Cooperative Path Planning for Multiple Agricultural Field Machines Performing Sequentially Dependent Tasks

Riikka Soitinaho

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 26.11.2018

Thesis supervisor and advisor:

D.Sc. (Tech.) Timo Oksanen

Aalto University
School of Electrical
Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Riikka Soitinaho

Title: Cooperative Path Planning for Multiple Agricultural Field Machines Performing Sequentially Dependent Tasks

Date: 26.11.2018 Language: English Number of pages: 8+118

Department of Electrical Engineering and Automation

Professorship: Control, Robotics and Autonomous Systems, ELEC3025

Supervisor and advisor: D.Sc. (Tech.) Timo Oksanen

Coverage path planning is the task of finding a collision free path that passes over every point of an area or volume of interest. In agriculture, the coverage task is encountered especially in the process of crop cultivation. Several tasks are performed on the field, one after the other, during the cultivation cycle.

Cooperation means that multiple agents, in this case vehicles, are working together towards a common goal. Several studies consider the problem where a single task is divided and assigned among the agents. In this thesis, however, the vehicles have different tasks that are sequentially dependent, that is, the first task must be completed before the other. The tasks are performed simultaneously on the same area. The literature review suggests that there is a lack of previous research on this topic.

The objective of this thesis was to develop an algorithm to solve the cooperative coverage path planning problem for sequentially dependent tasks. A tool chain that involves Matlab, Simulink and Visual Studio was adapted for the development and testing of the solution. A development and testing architecture was designed including a compatible interface to a simulation and a real-life test environment. Two different algorithms were implemented based on the idea of computing short simultaneous paths at a time and scheduling them in real-time.

The results were successfully demonstrated in a real-life test environment with two tractors equipped with a disc cultivator and a seeder. The objective was to sow the test area. The test drives show that with the algorithms that were developed in this thesis it is possible to perform two sequentially dependent agricultural coverage tasks simultaneously on the same area.

Keywords: CPP, mission planning, route planning, centralized planning, simultaneous cooperation, sequentially dependent tasks, field robotics, agriculture, tractor

AALTO-YLIOPISTO
SÄHKÖTEKNIIKAN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Riikka Soitinaho

Työn nimi: Reitinsuunnittelu määrätyssä järjestyksessä tehtäville peltotöille usean työkoneen yhteistyönä

Päivämäärä: 26.11.2018 Kieli: Englanti Sivumäärä: 8+118

Sähkötekniikan ja automaation laitos

Professuuri: Säätötekniikka, robotiikka ja autonomiset järjestelmät, ELEC3025

Työn valvoja: TkT Timo Oksanen

Työn ohjaaja: TkT Timo Oksanen

Kattavassa reitinsuunnittelussa yritetään löytää polku, jonka aikana määritelty ala tai tilavuus tulee käytyä läpi niin että alueen jokainen piste on käsitelty. Maataloudessa tämä tehtävä on merkityksellinen erityisesti peltoviljelyssä. Useita peltotöitä suoritetaan yksi toisensa jälkeen samalla alueella viljelyvuoden aikana.

Useissa tutkimuksissa käsitellään yhteistyönä tehtävää reitinsuunnittelua, jossa yksi tehtävä on jaettu osiin ja osat jaetaan useiden tekijöiden kuten robottien kesken. Tässä diplomityössä peltotyökoneilla on kuitenkin omat erilliset tehtävänsä, joilla on määrätty järjestys, eli niiden suorittaminen riippuu työjärjestyksestä. Työkoneet työskentelevät samanaikaisesti samalla alueella. Diplomityössä tehty kirjallisuuskatsaus viittaa siihen, että vastaavaa aihetta ei ole aiemmin tutkittu.

Tämän diplomityön tavoitteena on kehittää algoritmi, jolla voidaan toteuttaa reitinsuunnittelu määrätyssä järjestyksessä tehtäville peltotöille usean peltotyökoneen yhteistyönä. Algoritmikehitystä ja testausta varten suunniteltiin yhtenäinen rajapinta, jolla algoritmia voitaisiin testata sekä simulaatiossa että todellisessa testitilanteessa. Algoritmikehityksessä käytettiin työkaluina Matlab, Simulink ja Visual Studio -ohjelmia. Työssä toteutettiin kaksi algoritmia, jotka perustuvat samaan ideaan: suunnitellaan kerrallaan kaksi lyhyttä samanaikaista polkua, jotka ajoitetaan reaaliajassa.

Algoritmeja testattiin todellisessa testiympäristössä kahden työkoneen yhteistyönä kun tavoitteena on kylvää koko testialue. Ensimmäinen työvaihe suoritettiin lautasmuokkaimella ja toinen kylvökoneella. Testiajot osoittavat, että diplomityössä kehitetyillä algoritmeilla voidaan ohjata kahden toisistaan riippuvaisen peltotyön toteutus samanaikaisesti samalla peltoalueella.

Avainsanat: CPP, tehtävänsuunnittelu, reitinsuunnittelu, keskitetty suunnittelu, yhtäaikainen yhteistyö, järjestyksestä riippuvat tehtävät, peltorobotiikka, maanviljely, traktori

Preface

I didn't always know what I want to be when I grow up. I had many plans along the years and no matter what I chose to do, I always had the support from my family. Mom, Dad, you were always interested in my studies and at the same time you always let me find my own way. I truly appreciate it. And when sometimes my studies and work would feel too exhausting, I had people around me who could always make me smile. Jaakko Soitinaho, Jaakko Kantojärvi, Ville Sahlberg and Enni Lehto, thank you for always being great friends and great inspiration.

While doing my thesis I was lucky to be a member of one of the greatest teams I have ever worked with during my studies and my career: Janna Huuskonen, Vili Väyrynen and Timo Oksanen. I feel I was also lucky to have this thesis topic in the first place. Timo Oksanen, I will always be grateful to you for giving me this opportunity, and not least because along the way I think I finally found what I want to do in my career in the future.

And most of all, Aleksi Turunen, you were always there for me when I had doubts about myself and my ability to do a great thesis. Your continuous support during the difficult times is beyond value and I don't know how to thank you enough.

Otaniemi, 26.11.2018

Riikka Soitinaho

Contents

Nomenclature

Acronyms and abbreviations

ADS	Automated driving system
API	Application programming interface
CPP	Coverage path planning
DGPS	Differential GPS
DLL	Dynamic-link library
DOP	Dilution of precision
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GTSP	The Generalized Travelling Salesman Problem
GUI	Graphical user interface
HDOP	Horizontal dilution of precision
HIL	Hardware in the Loop
IP	Integer programming
KKJ	The national grid coordinate system in Finland
MPC	Model predictive control
NMEA	The National Marine Electronics Association
NP	Non-deterministic polynomial-time
PRN	Pseudorandom noise
RTK GPS	Real-time Kinematic GPS
SAE	The Society of Automotive Engineers
SLDRT	Simulink Desktop Real-Time
TCP	Transmission Control Protocol
TSP	The Travelling Salesman Problem
UDP	User Datagram Protocol
UTC	Coordinated Universal Time
VRP	The Vehicle Routing Problem
WGS	The World Geodetic System
YKJ	The uniform coordinate system in Finland

Terms used

Coverage	To cover something entirely as in visiting every point of an area of interest.
Field	An area of open land that is used for cultivation purposes.
Headland	A part of a field where field machines (for example tractors) make turns.
Implement	An agricultural tool which is connected to a tractor.
Line segment	(Or segment) A line that has no curvature and is bounded by two distinct end points.
Mainland	The main part of a planted field bounded by the headland.
Mission	An overall objective to complete one or several related tasks. A mission is considered complete when all the individual tasks are complete.
Path	A defined, traversable sequence of line segments that connects two points via a variable amount of other waypoints.
Polyline	A continuous line composed of one or more connected line segments.
Servicing	Emptying or refilling the tanks of an implement.
Swath	One continuous path which a tractor that is equipped with an implement can drive with the implement in working position without making a turn.
Task	One work phase to be performed by a field machine (for example a tractor) on the field, for example harrowing or sowing.
Waypoint	A set of coordinates that define a point in physical space. An intermediate point on a line of travel, or a point where course is changed. In this thesis, a waypoint is not only a coordinate point but contains other data as well.

1 Introduction

According to estimations by the United Nations (UN) in 2017 [1], the world population has grown by one billion inhabitants over the last twelve years. The current projections indicate that the population continues to grow today and in the near future, albeit with a decreasing rate. While the population is growing, the amount of arable land per person is in decline. According to the Food and Agriculture Organization (FAO) of the United Nations, arable land per person has decreased from estimated 0.38 ha in 1970 to approximately 0.19 ha in 2015 [2]. However, at the same time, yield per ha of cropped land has increased. According to FAO data, the world average cereal yield per ha has increased from 1.82 tonnes in 1970 to 3.97 tonnes in 2015 [3].

Today, as one of many things, the efficiency of food production is highly dependent on machinery. Advances in technology have given their contribution to the efficiency of agricultural production: mechanization and automation have significantly increased both productivity and convenience during the last century and recent years [4]. Increasing production efficiency is beneficial also from the practical point of view, including increasing output and fewer hours of human labour.

Coverage path planning is a central task in agricultural field operations such as tillage, planting, cultivation and harvesting. Both planning the path and steering the vehicle have traditionally been carried out by a human operator based on years of experience. Today, also commercial automation such as guidance systems exist to assist the human operator. These systems tend to use GPS for locationing [5], since the accuracy of modern GPS technology makes it possible to locate the vehicles within centimetres in real time.

1.1 Central Concepts

Coverage path planning is the task of finding a collision free path that passes over every point of an area or volume of interest. The ways to solve this task are numerous, but only a method that can provably cover the entire target region is called complete. In many applications this is indeed the desired outcome: to see, to fill, to paint, to harvest (and so on) the entire area of interest. Of course, the constraints for every application are different: the path needs to be determined so that all the manoeuvres are feasible.

In agriculture, the coverage task is encountered especially in the process of crop cultivation. During the cultivation cycle, the field is processed throughout several times. One way of traversing the entire field is to simply decide a driving direction and then drive back and forth across the field along parallel driving lines (sometimes called swaths). Optimally, when the swath centrelines are exactly one working width apart, the field is covered entirely with no gaps and no overlaps.

Today, in commercial cultivation, the tasks are usually operated with agricultural machines such as tractors. Modern tractors are increasingly equipped with assisting technologies such as guidance systems that allow the vehicle to be guided with GPS points and other necessary information. A path to cover the field plot can be entered to the guidance system that navigates the given route through the field.

Many tasks are carried out on the field during the year, including ploughing, harrowing, seeding and harvesting. The tractors are equipped with different implements to perform different tasks. Some of the tasks, such as harrowing and seeding, can be performed immediately one after the other. As well as multiple vehicles performing a common task, enabling parallel operation of different tasks can reduce the total completion time of the operation.

1.2 Research Framework

The applications of coverage planning are various. Such algorithms have been presented for autonomous lawn mowing [6], vacuum cleaning [7], spray painting [8], pocket machining [9], additive manufacturing such as 3D printing [10], imaging purposes like terrain reconstruction [11], demining [12], various surveying [13] and inspection purposes [14], harvesting [15] [16], as well as other agricultural field operations [17] [18]. There are aerial [19], underwater [20], as well as ground based applications. While many studies consider 2D surfaces, some address variation in terrain elevation [21] or coverage in 3D [22].

In agricultural applications, some special considerations apply when planning the coverage of a field. The vehicles are often nonholonomic and therefore the possible manoeuvres are limited. When the tractor is equipped with an implement, reversing is not possible while the tool is in use. With the tool in the working position, the curvature of the path is limited as well. Likewise, the passes (or swaths) are sometimes traversed in a special sequence instead of simply proceeding to the adjacent one.

In [23], the complete coverage of an agricultural field was studied including minimising the overlap between subsequent passes. In [16], a coverage plan for an automated harvester minimised the number of passes to process the field. In [24], the traversal sequence of the swaths was optimised to minimise the non-working distance of the field machines. These field work patterns called B-patterns were introduced in [25]. In contrast to conventional field work patters, B-pattern sequences are algorithmically optimised according to chosen criteria. The benefits of using B-patterns were further studied in [26].

In [27], two approaches were used to solve the coverage of agricultural fields. In the first approach, in addition to using a trapezoidal decomposition, a search for the best driving direction was developed. The second approach, inspired from *model predictive control* (MPC), uses a search horizon to limit the search space for the best possible route. To search for the best route, a simulation is made at the end of each swath. Only the first swath of the best route is applied, and then the process is repeated. With this approach, instead of driving only straight swaths, also following the field edges is possible.

The problem of optimising the swath orientation was studied in [28] as well, while also regarding the field boundaries. An objective function was used for economic evaluation of the swath and margin configuration. However, only straight swaths and fields with relatively simple geometry were considered.

In [29], a custom cost function was applied to find the optimal solution of decomposition and subregion driving direction for the coverage of agricultural fields.

The cost function considered the cost of different types of headland turns. In [21], a similar approach was applied to non-planar fields.

In [30], a genetic algorithm was used to optimise the visiting sequence of the subregions of a field. First, the swaths were generated in a desired angle and then grouped into subregions. Headlands were generated for the field and any obstacles. After optimising the visiting sequence, the final path was generated.

The challenges caused by variation of slope were further studied in [31], and a new approach was developed that ensures full coverage regardless of the field topology. An approach to evaluate the efficiency of coverage algorithms for varying elevation terrain was developed, and the efficiency of the developed coverage approach was demonstrated in both simulation and real-life. Another approach on optimising the field coverage was presented in [32], where a new swath traversal pattern was combined with an optimal decomposition of a complex field.

Some scenarios of cooperation have been studied as well. In [33], a method of developing a platooning system for autonomous agricultural vehicles was proposed. The system enables an autonomous tractor to follow a leading tractor at an offset, including during turning in the headlands. An approach for path planning for supporting transport units was presented in [34], where the support unit cooperates with a primary unit such as a combine harvester. Both in-field and inter-field paths were generated, while optimising time or travelled distance.

A coverage path planning approach for multiple agricultural vehicles was developed in [35], where the swaths were divided and assigned among available vehicles. A simulated annealing algorithm was used for optimisation of criteria such as distance, time and input cost.

1.3 Objectives of the Thesis

Whereas the general coverage path planning problem has been widely studied, regarding different applications and therefore different constraints and requirements, there seems to be a lack of research where multiple coverage path planning tasks are sequentially dependent and performed simultaneously on the same area. This possibility would be particularly useful in cultivation, where during the cultivation process, many individual coverage tasks are to be performed on the same field. It is not unlikely that this option could also be beneficial elsewhere. Therefore, this thesis aims to study and demonstrate if

> *... with a simple algorithm, is it feasible to simultaneously perform sequentially dependent CPP tasks on the same area of interest?*

To answer this question, two tasks were identified. First,

> *... from the previous literature, to conclude an applicable method of coverage path planning for a single vehicle,*

and further,

> *... to develop a novel and applicable method of extending the single vehicle method to enable parallel cooperation when individual CPP tasks are sequentially dependent.*

The main objective of this thesis is to provide a new algorithm for cooperative coverage path planning when multiple autonomous field machines are simultaneously performing sequentially dependent tasks on the same area of interest.

The results are to be demonstrated in an empirical test set-up with two agricultural vehicles. Both vehicles are equipped with an implement that is used to perform the tillage tasks. The test consists of two tasks with the following precedence. The first task is performed by a semi-autonomous tractor that is equipped with a harrow. The second task is performed by a fully autonomous tractor that is equipped with a seeder.

Due to the challenging nature of the real-life test set-up, including two several tonne vehicles and having several hour completion time, it is necessary to conduct preliminary tests in a simulated environment. Therefore, the implementation of a simulation environment is considered a part of this thesis. Moreover, the purpose of the simulation is twofold: first, to validate the real-time operation of the vehicle guidance software including the interface to the vehicles, and second, to validate the algorithm itself. Validating the algorithm in a simulation both ensures a safer operation during the real-life tests as well as provides the possibility of generating a broader set of test environments outside the real-world constraints. Likewise it is obvious that using a simulation environment can save a considerable amount of time. Otherwise the simulation environment is to be modelled after the empirical test set-up, including kinematics and parameters of the two vehicles.

Following these requirements, the objective of this thesis is structured as follows. The main objective of this work is to

(1) provide and test a new algorithm for cooperative CPP where sequentially dependent tasks are performed simultaneously on the same region of interest.

The main objective is complemented with auxiliary tasks concerning the testing and validation of the algorithm, including

(2) a real-time hardware-in-the-loop simulation,

(3) a non-real-time simulation and

(4) testing in a real-life test environment.

The primary purpose of the real-time HIL simulation is to verify the correct operation of all interfaces to the vehicles, whereas the purpose of the non-real-time simulation is to examine the performance of the algorithm.

1.4 Structure of the Thesis

The structure of this thesis is as follows. The main objective, as stated before, is to provide an algorithm for cooperative coverage path planning for multiple vehicles performing sequentially dependent tasks on the same region of interest. This task is essentially twofold. First, a method of solving the general region filling problem is needed. Second, the overall task includes at least two region filling tasks that are sequentially dependent and to be performed simultaneously on the same region of interest. The subject is first studied through previous literature on the topic. Then, drawing from previous advances in the fields, a novel approach is developed and introduced.

The problem itself as well as the application domain are multifaceted and thus a short introduction to all central concepts is given. The specific area of application, agricultural field operations and autonomous driving, is briefly introduced in chapter 2. A research review on state-of-the-art path planning and coverage path planning is given in chapter 3. A description of the mission for this thesis and related key definitions is presented in chapter 4. The approach that was implemented in this thesis to solve the cooperative coverage path planning problem is described in detail in chapter 5. The approach was planned to be validated first in simulation and then in a real-life test environment. The details of the test cases for both validation methods are described in chapter 6 and chapter 7, respectively. The results and their evaluation are reported in chapter 8 and chapter 9. Finally, chapter 10 concludes this thesis with a summary and propositions for future work.

Three theses, including this one, were conducted as a part of a project titled *Hybrid Autonomous and Augmented Agricultural Tools* (HAUGE) funded by the Academy of Finland. The project is focused on the cooperative scenario involving two autonomous tractors. The tractors perform their individual field operation tasks simultaneously on the same field. All three theses have their individual topics that contribute to the project goal: this thesis work provides the cooperative path planning algorithm, while the other two provide the turning path algorithm for the headlands and an augmented reality application to serve as a user interface for the human operator. The other two theses and their contributions are mentioned in this study where necessary.

2 Background

A brief background to the application area and the relevant technology is presented in this chapter. A short introduction to agriculture and cultivation is given in section 2.1. The concept of level of automation is introduced in section 2.2. The global positioning system technologies for absolute positioning outdoors are introduced in section 2.3 and different coordinate systems are briefly described in section 2.4.

2.1 Agriculture and Cultivation

Advances in technology have given their contribution to the efficiency of agricultural production: mechanization and automation have significantly increased both productivity and convenience during the last century and recent years [4]. Many tasks are carried out on a planted field during the year, including ploughing, harrowing, seeding and harvesting.

The tasks are usually operated with agricultural machines such as tractors. The tractors are equipped with different implements to perform different tasks. Both planning the path and steering the vehicle have traditionally been carried out by a human operator based on years of experience. Today, also commercial automation such as guidance systems exist to assist the human operator. These systems tend to use GPS for locationing [5], since the accuracy of modern GPS technology makes it possible to locate the vehicles within centimetres in real time.

The tractors are equipped with implements to perform the field operation tasks such as ploughing. The implements are mounted or towed. The tractor is used for pulling power and the implement is used for performing the cultivation task. The implement has an impact on the tractor movement, for example the turning radius is often limited due to the constraints regarding the implement. A hitch mounted implement can be lifted up from the working position so that turns can be made with less restrictions.

The process of planting crops in the prepared soil is called sowing. Secondary tillage is performed on the field area before the crops can be sown. Secondary tillage is done to improve the seedbed for sowing and to provide optimal conditions for germination, including the amount of moisture and mixing of the soil. After secondary tillage the seedbed is level and uniform for sowing. Also weeds are destroyed with secondary tillage.

For example, a disc cultivator can be used for secondary tillage and a combined seed drill can be used for sowing the crops. The depth of the secondary tillage is the depth where the crop seeds are placed. With a seeder the seeds are sown at even distances. The soil can be fertilized at the same time when sowing with a combined seeder fertilizer.

2.2 Level of Automation of Autonomous Vehicles

The SAE International recommended practice J3016 [36] provides a six-level definition for the level of automation for motor vehicle driving automation systems that perform

part or all of the dynamic driving task. The definition addresses particularly on-road vehicles, however, regarding the definition of the level of automation it is mostly applicable to other types of automated ground vehicles as well.

The dynamic driving task includes the operational and tactical aspects of the driving task. The operational aspects include low-level functions such as accelerating, steering, and monitoring the vehicle and the surroundings. The tactical aspects include e.g. turning and responding to events. Then, a driving mode is a type of driving scenario with typical dynamic driving task requirements, such as high speed cruising, expressway merging or parking assistance for an on-road vehicle.

The automation level of a driving system is specified based on (1) the amount of necessary involvement of a human driver in the dynamic driving task and (2) system capability i.e. amount of available driving modes. The automation level taxonomy ranges from level 0 "no driving automation" to level 5 "full driving automation".

A level 0 system is a "No Automation" driving system where the human driver is in complete control of the vehicle. The system does not intervene with driving, but it may include warning or intervention features. A level 1 system is a "Driver Assistance" driving system where the human driver is in control of the vehicle, but the system may modify the speed and direction of the vehicle according to a driving mode. A level 2 system is a "Partial Automation" driving system where the human driver is responsible for monitoring the environment and performing corrections accordingly, but the system is capable of autonomously performing some driving modes including full control of speed and steering of the vehicle. A level 3 system is a "Conditional Automation" driving system where the monitoring of the environment is mainly performed by the system itself, but a human driver must be present to perform any necessary corrections. A level 4 system is a "High Automation" driving system where the presence of a human driver is no longer needed, but the system capabilities are limited to specific conditions or driving modes. Finally, a level 5 system is a "Full Automation" driving system where the system is capable of full-time controlling the vehicle in any on-road conditions manageable by a human driver.

The summary of classifying features for different driving automation levels can be seen in the J3016 summary table in figure 1. Automation levels 3 to 5 correspond to Automated Driving Systems (ADS), where the system itself is capable of monitoring the driving environment. To be considered an ADS, the system needs to execute steering, acceleration and deceleration autonomously and have some driving modes available as well.

SAE level	Name	Narrative Definition	Execution of Steering and Acceleration/ Deceleration	*Monitoring* of Driving Environment	Fallback Performance of *Dynamic Driving Task*	System Capability *(Driving Modes)*
***Human driver* monitors the driving environment**						
0	No Automation	the full-time performance by the *human driver* of all aspects of the *dynamic driving task*, even when enhanced by warning or intervention systems	Human driver	Human driver	Human driver	n/a
1	Driver Assistance	the *driving mode*-specific execution by a driver assistance system of either steering or acceleration/deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	Human driver and system	Human driver	Human driver	Some driving modes
2	Partial Automation	the *driving mode*-specific execution by one or more driver assistance systems of both steering and acceleration/ deceleration using information about the driving environment and with the expectation that the *human driver* perform all remaining aspects of the *dynamic driving task*	**System**	Human driver	Human driver	Some driving modes
***Automated driving system* ("system") monitors the driving environment**						
3	Conditional Automation	the *driving mode*-specific performance by an *automated driving system* of all aspects of the dynamic driving task with the expectation that the *human driver* will respond appropriately to a *request to intervene*	System	**System**	Human driver	Some driving modes
4	High Automation	the *driving mode*-specific performance by an automated driving system of all aspects of the *dynamic driving task*, even if a *human driver* does not respond appropriately to a *request to intervene*	System	System	**System**	Some driving modes
5	Full Automation	the full-time performance by an *automated driving system* of all aspects of the *dynamic driving task* under all roadway and environmental conditions that can be managed by a *human driver*	System	System	System	**All driving modes**

Figure 1: SAE International J3016 level of driving automation summary table [37].

2.3 Absolute Positioning Outdoors

Global positioning systems are an essential technology in precision agriculture and field robotics for both research and practical purposes. [38] In outside environments, a GPS receiver is often used as a primary source of absolute location for fully autonomous vehicles. For precision operations such as cultivation, centimetre level accuracies can be obtained with advanced GPS technologies. [5]

Global Positioning System

A satellite navigation system is a system that uses satellites to provide geo-spatial positioning information (longitude, latitude, altitude). A satellite navigation system that has global coverage is often called a global navigation satellite system (GNSS). The Global Positioning System (GPS) is a GNSS that was developed by the U.S. Department of Defence in the 1970s. [39] Other GNSS systems include the Russian GLONASS and Galileo by the European Union.

In the original GPS system design a GPS satellite transmits a radio signal composed of two carrier waves L1 (1575 MHz) and L2 (1227 MHz), two pseudorandom noise (PRN) ranging codes, and a navigation message. The two ranging codes, a signature pattern of 1023 plus and minus signs, identify the satellite that transmitted the signal. The precise code (P code) is only for military access, whereas the coarse/acquisition code (C/A code) is also available for civilian use and published in a public database. [40] A modernization program for the GPS system is under implementation. The legacy civil signal L1 C/A will be accompanied with new civil signals L2C, L5 and L1C. As of October 18, 2018, the L2C and L5 signals are only

pre-operational and being broadcast from 19 and 12 satellites, respectively. [41]

The nominal constellation of 24 satellites is to ensure that at least four satellites are visible from almost any point on the Earth. For the past few years, 31 satellites have been operational. [41] To determine its location, a GPS receiver tracks multiple visible satellites simultaneously. Each satellite carries a precise atomic clock to ensure the frequency of the carrier and the stability of the signal. The receivers, however, are often equipped with less expensive clocks whose drift is unknown. With the time stamp included in the GPS signal, the receiver is able to determine an approximate transmission delay from the satellite. The percieved distance to the satellite can de determined from the travel time of the signal, however, it is not the true distance to the satellite due to the clock error, and is therfore often called the *pseudorange*. Because of the unknown error in the receiver clock, three satellite-receiver distances are not enough to determine the location of the receiver, but at least four satellites are needed to account for the receiver clock offset. [40] The accuracy of the location information is increased with the number of simultaneously tracked satellites included in the location calculation.

In addition to various error sources and biases, also the satellite constellation geometry affects the GPS position accuracy. The constellation geometry represents the relative positions of the tracked satellites as seen by the receiver. In general, a more desirable satellite geometry is when the satellites are spread further apart. A value called dilution of precision (DOP) is used to measure the influence of the satellite geometry on positioning accuracy. A lower DOP value indicates a better geometry. [39]

Positioning Modes

The GPS positioning can be performed as either point positioning or relative positioning. In point positioning, only one GPS receiver is used. If the receiver can see at least four satellites, it can determine its position as discussed before with either of the PRN codes from all the visible satellites. Horizontal accuracy of around 20 meters is obtainable from the C/A code available for civilian use. [39]

Relative GPS positioning is used for high accuracy applications such as precision navigation. Relative positioning makes use of two receivers tracking at least four common satellites. The other receiver serves as a reference station and its exact location is known. For the other receiver, often called rover, the coordinates are unknown: it can be in continuous movement as well. [39]

In differential GPS (DGPS), the reference station is used to calculate and transmit pseudorange error corrections to the rover. Within certain proximity to the reference station, the errors in the measured pseudoranges at the reference station and the rover are practically the same. The reference station transmits the error corrections to the rover and the rover uses the error corrections to correct its own pseudorange measurements. The rover location is then computed using the corrected pseudoranges. With DGPS, meter level real time accuracy can be obtained. [39]

However, high-precision navigation often demands a centimetre-level accuracy in real time. Real-time kinematic GPS (RTK GPS), sometimes also referred to as carrier-

phase enhancement GPS (CPGPS), is based on carrier-phase relative positioning. An RTK GPS system typically consists of a roving station and a stationary reference station in a known location tracking the same satellites as the mobile receiver. The reference station relays correction data to the roving station. Using measurements of the signal carrier wave phase together with correction transmission allows obtaining centimetre-level positioning accuracy in real-time [42].

Message Protocol

A protocol is needed to interpret the GPS data from the receiver. The NMEA 0183 is a voluntary industry standard by the National Marine Electronics Association (NMEA). The NMEA 0183 defines a data protocol for communication between marine instrumentation including GPS receivers. The messages are called *sentences*: GPS fix data is transmitted in the GGA sentence. The GGA sentence contains the time, position and solution related information, including latitude, longitude and altitude of the receiver and the time of position in UTC. In addition, the accuracy and solution related data contains, among others, fix quality, number of tracked satellites for the solution and horizontal dilution of precision (HDOP). The fix quality is indicated with integer values, including [39]

0 = fix not available

1 = point positioning with P-code

2 = DGPS with C/A-code

3 = point positioning with C/A-code

4 = integer RTK GPS

5 = float RTK GPS

2.4 Coordinate Systems and Geodetic Datums

A point on Earth's surface can be expressed as map coordinates that are given either as geographical or horizontal coordinates. Geographical coordinates are defined as values of latitude and longitude, latitude being the north-south position and longitude being the east-west position of a point. Both latitude and longitude are typically expressed in degrees. Horizontal coordinates, on the other hand, are defined as values of x and y coordinates. The term easting refers to the eastward measured distance, or the x coordinate, whereas the term northing refers to the northward measured distance, or the y coordinate. Easting and northing are commonly measured as distances from some specified location in the given coordinate system. These distances are based on a map projection and do not generally match the distances as measured on the surface of the Earth.

A coordinate system and a set of reference points that are used to locate points on the Earth is called a geodetic datum. A geodetic datum can be used to translate

two-dimensional map positions to real positions on Earth. The World Geodetic system is a global standard coordinate system for the Earth, the latest revision of which is WGS 84. The GPS uses WGS 84 as the reference coordinate system.

There are various localised datums as well. Using different datums will result in a different translation, and this difference between datums is often referred to as datum shift. A localised datum may give a more accurate representation of the area of its coverage than a global datum.

The National Grid Coordinate System

The national grid coordinate system (*fin. Kartastokoordinaattijärjestelmä, KKJ*) is a geodetic datum in Finland based on Gauss-Krüger projection. There are two coordinate systems to define and represent coordinates according to the KKJ: the basic coordinate system and the uniform coordinate system. [43]

For the basic coordinate system the area of Finland is divided into six bands of equal width, four centremost of which are commonly used [43]. The central meridians of the six bands are 18, 21, 24, 27, 30 and 33 degrees east longitude, and they are called KKJ0-KKJ5 respectively. The easting of a location is measured as its distance from the central meridian, whereas the northing is determined by the distance to the equator.

The Uniform Coordinate System

Another coordinate system in the national grid coordinate system is the uniform coordinate system (*fin. Yhtenäiskoordinaatisto, YKJ*). In the uniform coordinate system, as opposed to the basic coordinate system, the area of Finland is presented as one band instead of four. The central meridian of YKJ is 27 degrees east longitude, which aligns with the third band of the basic coordinate system. [44] To avoid negative coordinate values the central meridian is given the value of 3 500 000 metres.

3 Coverage Path Planning

The objective of coverage path planning is to determine a collision free path that passes over every point of an area or volume of interest. Coverage path planning can be considered a special case of path planning, where the task is to find a collision free path between two locations within an environment. This chapter presents a literature review of the past and current research on coverage path planning and several applications. First, a set of general definitions is given in section 3.1. A brief introduction to path planning is given in section 3.2. A literature review of coverage path planning and coverage path planning in agriculture is presented in sections 3.3 and 3.4, respectively. A brief summary of the findings is given in section 3.5.

3.1 General Definitions

The topic of coverage path planning involves several commonly used concepts and definitions. A set of general definitions is given to unify and clarify the presentation of the coverage path planning literature, and later, the description of the implementation details of the coverage path planning algorithms developed in this thesis. These definitions consist primarily of definitions used in computational geometry and graph theory and they are used throughout the literature.

Line Segments and Polylines

A *line segment* is a line bounded by two distinct endpoints. A line segment can be either *closed*, *open* or *half-open* depending on whether it includes both endpoints, excludes both endpoints or includes the other and excludes the other. A *polygonal chain* is a sequence of pairwise connected line segments. Sometimes a polygonal chain is referred to as a *polygonal curve*, a *piecewise linear curve* or a *polyline*, as is done in this thesis. If any two line segments of the polyline intersect, the polyline is *self-intersecting*. Otherwise it is often called a *simple* polyline. A polyline whose first and last line segment are connected is called a *closed* polyline. A closed polyline is equal to a *polygon*. A polyline is called monotone with respect to a straight line l if every line that is orthogonal to l intersects the polyline at most twice. The same definition of monotone holds for polygons as well.

Concave and Convex Polygons

A simple *polygon* is a two dimensional shape whose edges are straight, non-intersecting, pairwise connected line segments that form a closed polyline. The line segments are often called edges and the point at which the edges are connected is called a vertex. A simple polygon is either *convex* or *concave*. Two angles are formed between any two adjacent edges of a polygon: one that is inside the polygon (an *interior* angle) and one that is outside the polygon (an *exterior* angle). A simple polygon is convex if all of its interior angles are smaller than 180 degrees [45]. A concave polygon is a polygon that is not convex. An example of a convex and a concave polygon are shown in figure 2a and figure 2b, respectively.

(a) A convex polygon (b) A concave polygon

Figure 2: A simple polygon is either convex or concave.

Polygon Offsetting

An offset polygon is a polygon that is constructed from another polygon by offsetting its edges. The offset edges are parallel to the corresponding edges in the original polygon at an equal distance d, where $d > 0$. Outwards offsetting and inwards offsetting are often termed separately as *outsetting* and *insetting*, respectively. A polygon and an inset polygon is shown in figure 3.

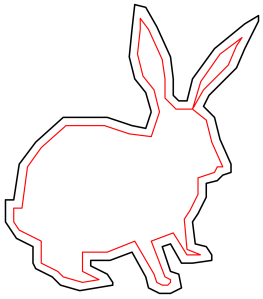


Figure 3: A polygon and an inward offset polygon.

Graphs, Walks and Paths

A graph is a pair $G = (V, E)$, where V is the set of vertices and E is the set of edges. An edge in the edge set E connects two vertices of the vertex set V, and the vertices connected by an edge are called *adjacent*. The degree of a vertex v is the number of edges at v. If all the vertices in V are pairwise adjacent, the graph G is called *complete*. [46]

A path is a graph $P = (V, E)$, where all the vertices in V are distinct and connected by E so that the maximum degree of any vertex is 2. A path whose endpoints are connected is called a cycle. A path P in a graph G is a subgraph of G. A walk in a graph is an alternating sequence of vertices and edges such that every edge of the walk connects the previous and the following vertex of the sequence. If all the vertices in a walk are distinct, it defines a path in the graph. If the endpoints of a walk are connected, the walk is closed (sometimes called a tour). [46]

In an undirected or directed graph, a Hamilton path is a path that visits each vertex of the graph exactly once. A Hamilton cycle begins and ends at the same

vertex. An Euler walk is a walk that visits each edge of the graph exactly once. A closed Euler walk is an Euler tour. [46]

The Travelling Salesman Problem

The Travelling Salesman Problem (TSP) is the problem of finding the shortest route to visit a set of customers in different locations so that each location is visited exactly once, returning to the initial location. Formally the TSP is defined in a graph G where each vertex represents a customer and each edge $e_i \in E$ is associated with a (travelling) cost c_i. The solution to the TSP is the Hamiltonian cycle in G with the smallest total (travel) cost. [47] In the generalized version of the TSP (GTSP) the vertices are partitioned into groups and the task is to find the shortest Hamiltonian cycle that visits each group of vertices exactly once.

The Vehicle Routing Problem

The Vehicle Routing Problem (VRP) is in many ways similar to the TSP, however, the customer demands and the vehicle capacities are considered in the VRP. The VRP is formally defined in a graph G as in the TSP: in addition, each vertex is associated with a demand and the capacity of each vehicle is defined. The solution to the VRP is to find a set of paths, one for each vehicle, visiting all the customers exactly once with a sufficient supply. [48]

Holonomic and Nonholonomic Systems

A system is holonomic if all constraints of the system are holonomic. A holonomic constraint depends only on the positional variables of the system. If a constraint depends on the time derivative of at least one of the variables and cannot be integrated to a constraint that involves none, then the constraint is nonholonomic. [49] A system is nonholonomic if it has at least one nonholonomic constraint.

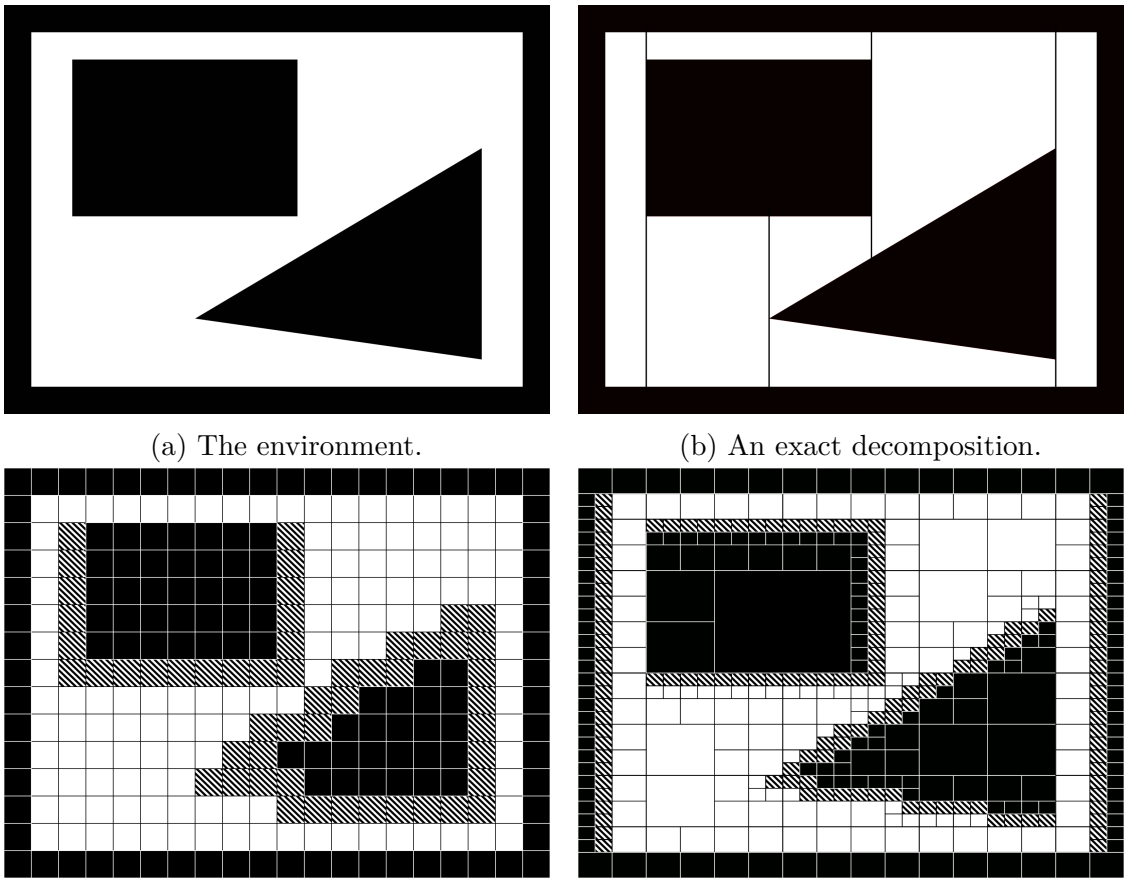
In practice, a nonholonomic vehicle is not able to move in all directions instantaneously. A car-like vehicle that is controlled with velocity and steering has only two degrees of freedom. Driving in a two dimensional plane, the dimension of its configuration space is three. Having a lower degree of freedom than the dimension of its configuration space, the car-like vehicle is under actuated. Therefore, it cannot move in all directions without an additional sequence of manoeuvres. [49]

3.2 Path Planning

Path planning is the task of finding a collision free motion between two configurations within an environment [50]. The terms *motion* and *configuration* are used to emphasize the implications of physical and kinematic constraints. For example, a vehicle is equipped with a tool. The solution to move the tool along a desired path in an environment is a suitable sequence of collision free vehicle configurations. Finally, for clarification, a path is considered to be purely a geometric matter. A *trajectory* is a path that considers the temporal aspect as well.

Path planning algorithms can be divided into three categories as in [51]: (1) *roadmap* methods, (2) *cell decomposition* methods and (3) *artificial potential* methods. In the roadmap methods, the connectivity of the collision free configurations is represented as a network of one dimensional curves. In the cell decomposition methods, the free configuration space is represented as a set of cells and an adjacency graph. In the artificial potential methods, the environment is represented as a potential field where the target configuration and the obstacles generate attractive and repulsive potentials. [51]

The cell decomposition methods can be further classified as approximate, semi-approximate and exact cell decompositions. A demonstrative environment is shown in figure 4a accompanied by a corresponding illustration of each previously introduced decomposition in figures 4b–4d. An empty cell is white, a mixed cell is grey and a full cell is black. The idea of decomposing a complex environment into more simple cells is often applied in coverage path planning as well. The different type of decompositions are described in more detail in section 3.3.



(a) The environment.

(b) An exact decomposition.

(c) A fixed decomposition.

(d) An adaptive decomposition.

Figure 4: Three types of cellular decompositions for the original environment in figure 4a. Adaptation based on presentation by Latombe in [51] and Siegwart et. al. in [52].

3.3 Coverage Path Planning

Sometimes, instead of planning a path from point A to point B, a set of points needs to be visited. The objective of coverage path planning is to determine a collision free path that passes over every point of an area or volume of interest. In robotics, this task has many applications such as autonomous lawn mowing [6], vacuum cleaning [7], spray painting [8], pocket machining [9], additive manufacturing such as 3D printing [10], imaging purposes like terrain reconstruction [11], demining [12], various surveying [13] and inspection purposes [14], harvesting [15] [16], as well as other agricultural field operations [17] [18]. There are aerial [19], underwater [20], as well as ground based applications. While many studies consider 2D surfaces, some address variation in terrain elevation [21] or coverage in 3D [22].

Some general requirements for coverage planning were proposed in an early study where the task was termed *region filling* [53]. As a general guideline, the proposition includes that the region filling solution

(1) must cover the entire area during overall travel

(2) with no overlapping paths,

(3) performing continuous and sequential operation

(4) while avoiding all obstacles and

(5) using simple motion trajectories and

(6) yielding an *optimal* path given the preconditions.

In reality, fulfilling these criteria is often a matter of prioritization between application specific real-life constraints.

The exact meaning of coverage is dependent on the application at hand. For example, for a lawn mowing robot the obvious objective is to cut all the grass. Then the coverage is dependent on the location and measures of the cutter rather than the dimensions of the robot itself. Likewise, for a surveying mission, the coverage is dependent on the beam or view of the sensing element such as a laser scanner or a camera. While the footprint of the tool determines the covered area, the kinematic constraints of the robot define the feasible trajectories to move the tool.

Obtaining optimal coverage is, nonetheless, not an easy task. Given a tool with fixed dimensions, finding the shortest coverage path has been proven to be NP-hard in general, even for a simple polygonal region [6]. Therefore, approximate or sometimes even heuristic solutions can be considered sufficient.

Having several application areas with distinct requirements and constraints, it is beneficial to introduce a classification of CPP algorithms on grounds of their significant qualities. Choset originally proposed a classification into four categories: heuristic, approximate, partial-approximate and exact cellular decompositions [54]. The basis of this classification is *completeness*, i.e. whether the approach can guarantee that the entire area is covered. A CPP algorithm is complete if it can be proven to cover

the area of interest entirely, while an approach that can not guarantee full coverage is not complete. In a heuristic approach, the overall exploration behaviour is produced by combining a set of simple rules. Relying only on simple rules, a heuristic approach is not complete. Another trivial example of a non-complete algorithm is a random travel inside the area boundaries: it may or may not complete the task in finite time.

Unlike heuristic approaches, the cellular decompositions are at least resolution complete. Choset made a division between approximate, semi-approximate and exact cellular decompositions. An approximate cellular decomposition can be considered complete up to a resolution. Divided into a fine grid, covering the area of interest equals visiting each grid cell. However, the grid is not an exact representation of the environment due to discretization, and therefore the approach is only resolution complete. In a partial-approximate decomposition the width of the cells is fixed, but the two connecting edges can have any shape. Finally, an exact cellular decomposition divides the region of interest into non-overlapping areas such that their union is exactly the original region.

Then, independently of its completeness, a CPP algorithm can be classified based on the availability of prior information of the environment [54]. Either the environment is fully known, the information of the environment is partial or nonexistent, or the environment evolves with time. If the environment is known and stationary, all planning can be performed off-line before initiating the execution of the task. If sufficient information is not available, it needs to be acquired while performing coverage on-line. Therefore this partition is often referred to as *off-line* and *on-line* algorithms, while the latter is sometimes called *sensor-based* due to utilising on-board sensors in gathering the necessary information of the environment.

Of course, full knowledge of the environment does not exclude the algorithm being capable of performing on-line. In a real-life scenario, perfect knowledge of the environment requires both a perfect map and a sensor capable of providing an accurate location with high precision.

In a real-life scenario, perfect a priori knowledge of the environment is not exactly enough to guarantee completeness. A high accuracy and precision sensor is required to provide location information as feedback. Even then, accounting for the noise and other error sources, optimal coverage of the region of interest is debatable.

Last, in addition to completeness and availability of prior information, Choset regards the issue of *time*. The duration of travelling the coverage path is not only a function of distance but also a function of speed, and turning is often more time-consuming than driving in straight lines. Therefore, to obtain better *time-to-completion*, one may consider minimizing the amount of turns as well [54].

On the grounds of their survey, Choset presumed that, at the time, most complete algorithms applied a cellular decomposition to obtain complete coverage. A cellular decomposition is a partitioning of the free space (the area of interest) into a set of smaller regions called cells. Used in path planning in general, cell decompositions are a means to enable motion planning in an environment. The union of all the cells equals the original environment in size but not always in resolution. Therefore, cellular decompositions are often classified as either exact or approximate. [54]

In an approximate decomposition the cells are required to have a predefined

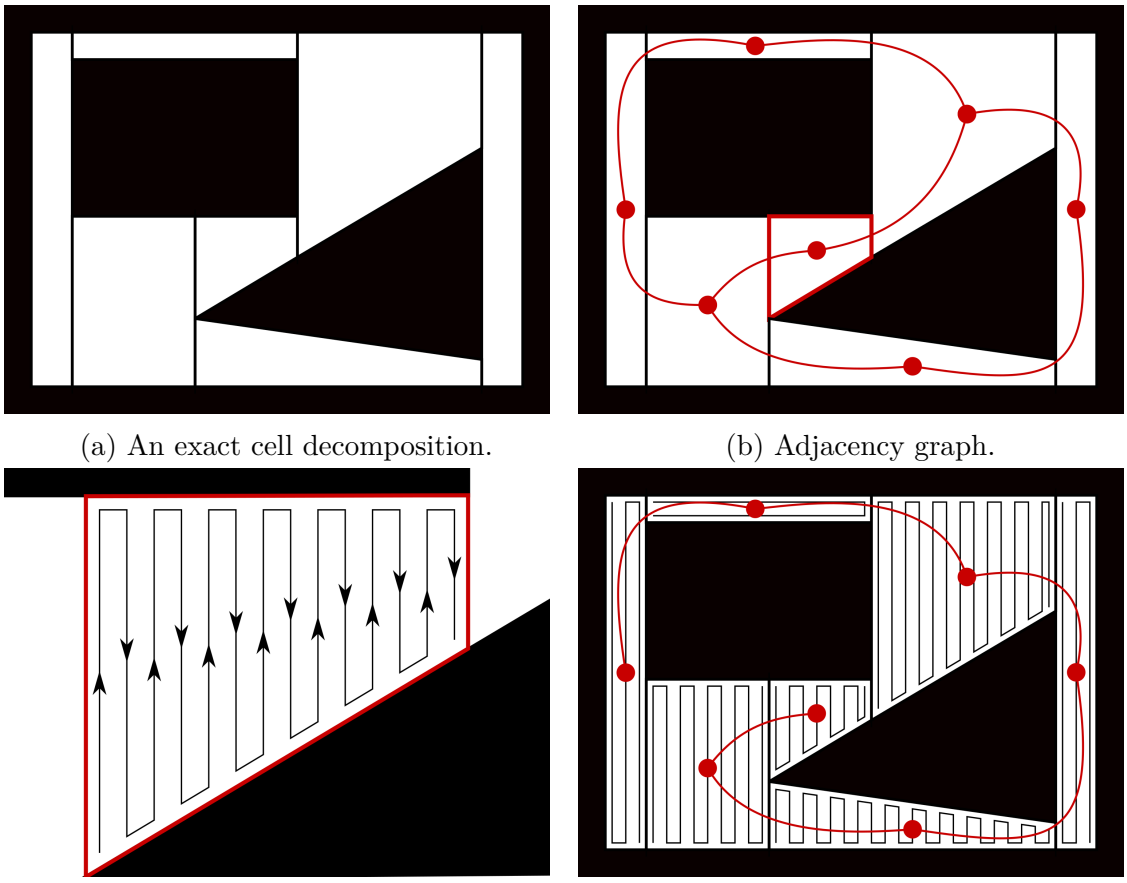
shape. Each of the cells are marked as either *empty*, *mixed* or *full*. [51] Any cell that is occupied by an obstacle is marked as full, whereas a mixed cell is not entirely free of obstacles. Sometimes the classification as mixed is omitted and the cell is marked as full instead.

An approximate cell decomposition can be further classified as a fixed cell decomposition or an adaptive cell decomposition. As the name suggests, in a fixed cell decomposition the cells are of fixed size and shape. This results in a discrete approximation, usually in the form of square or rectangular cells. [52] Even with the loss of information, many times a fixed decomposition is a reasonable representation of the region of interest.

To overcome some of the challenges of a fixed decomposition, adaptive decompositions allow the cell size to be locally adjusted [51]. One way of enabling adaptation, as presented in [52], is to recursively divide every non-uniform cell into four equal-sized cells until a predefined treshold of cell-size is reached. That is to say, every mixed cell is partitioned into four smaller cells, whereas empty and full cells are left as is.

Finally, an exact cell decomposition is a partition of a region into a set of non-overlapping cells whose union precisely describes the original region. Since no information is lost in the decomposition, it is a lossless representation of the environment. One way of constructing an exact decomposition is to sweep an axis-parallel line across the region. Once the line encounters a vertex of an obstacle, sometimes called a critical point, the closing edge of the previous cells and the opening edge of the following cells is determined. Generally a graph or a similar data structure is used to describe the connectivity among the cells in a cell decomposition.

An example of an exact cell decomposition is the trapezoidal cell decomposition where the environment is decomposed into trapezoidal cells [51]. The coverage of each cell is simple because all the cells are trapezoids [54]. An example of coverage path planning with a trapezoidal cell decomposition is shown in figure 5. Complete coverage of the environment is achieved by visiting every cell in the decomposition and covering them individually. The path to visit every cell is planned based on the adjacency graph. The adjacency graph is shown in figure 5b and a possible visiting path in figure 5d with the individual back-and-forth coverage of each cell.



(a) An exact cell decomposition.

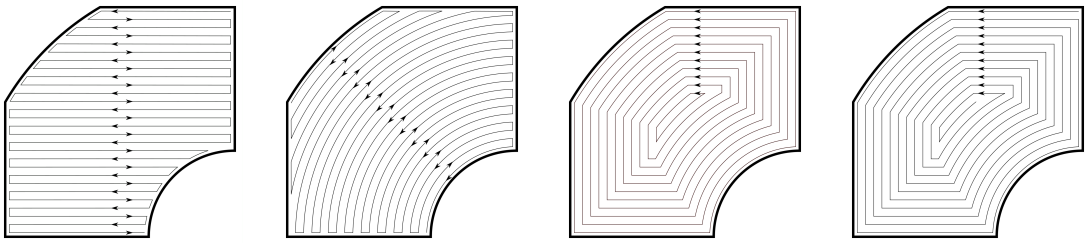
(b) Adjacency graph.

(c) Individual cell coverage.

(d) Complete coverage.

Figure 5: Coverage path planning with a trapezoidal decomposition. Adaptation based on description in [51] and [54].

The coverage of the individual cells in a trapezoidal cell decomposition is easy in the sense that simple shapes can be processed with trivial back-and-forth movements. One solution is to compute back-and-forth paths that are parallel to the longest edge of the polygonal cell. Different methods of covering the individual cells (or more complex areas as well) are shown in figure 6. The cell in the examples is not convex, however, in this case it is possible to find a reasonable axis-parallel filling even when the cell is concave.



(a) Axis-parallel. (b) Edge-following. (c) Contour-parallel. (d) Spiral.

Figure 6: An example of different methods of simple region filling identified from the literature.

These concepts are used widely in the coverage path planning research and the literature on coverage path planning is vast. Since the earliest research, many new aspects and applications have been discovered. All applications have their own objectives and constraints and therefore not all approaches are suitable for all applications. However, the relevant ideas provide valuable insight to the challenges of coverage planning and their solutions. Many essential details can be adapted for different purposes. The objective of this literature review is to find and evaluate relevant research on coverage planning for the purposes of this thesis, and at the same time, to present the reader with some of the history and applications of the coverage path planning research. After a brief introduction, the following review is divided into general and agriculture specific research. Furthermore, to highlight the differences between single robot and multiple robot approaches, these approaches are presented separately.

Single Robot Approaches

In addition to introducing many of the fundamental principles of coverage path planning, also one of the earliest coverage solution was presented in [53]. The simple coverage strategy consists of straight parallel paths travelled back and forth. Once an obstacle is detected online the coverage is continued on its other side until reaching its opposite end. By tracking the obstacle boundary back to the detection point, the coverage is then resumed on the other side. A simulation result was presented to demonstrate the solution.

Another early approach was presented in [7], where the coverage of a room with obstacles was studied for a cleaning robot. For a simple room, no prior information of the layout was given to the robot, whereas for a complex room the layout as well as the travel path were input in advance. For the simple room, a back and forth coverage path is planned online parallel to a wall based on distance information from ultrasonic sensors. The solution was tested on a small cleaning robot in two different rooms.

In [55], an approximate cell decomposition approach was used to find a coverage path in an environment with obstacles. The approach is based on *distance transform* used in path planning where a *distance wave* originating at the target location is propagated through the free space. Then from any initial location in the environment,

the shortest path is along the steepest descent. The distance transform can be used as is to plan a coverage path, however, it results in an excessive amount of turns. To provide a feasible coverage plan, the distance transform was extended to consider the proximity of obstacles. For each cell in the environment, the extended transform assigns a value based on the weighted sum of the distance to the target location and the *discomfort* of moving close to obstacles. The new approach results in a contour parallel coverage path decreasing the amount of turns. The approach was demonstrated in a simulation and with a mobile robot where dead reckoning with correction from ultrasonic sensors was used for navigation.

In [56], a new exact decomposition approach based on a line sweep was developed to allow a different sweep direction in every subregion of a complex concave environment with obstacles. The environment is assumed polygonal and therefore all subregions obtained with a line sweep are polygonal as well. A measure called region *altitude* was defined as a function of the sweep direction. Given the region orientation, the altitude is the diameter between the highest and the lowest point of the polygon. The optimal decomposition is determined by the sweep direction that minimizes the sum of the subregion altitudes. An algorithm is presented where an initial decomposition is obtained by dividing the environment into cells with several line sweeps and extending any concave edges to intersect the environment boundary. With dynamic programming, the cells are combined into larger subregions and assigned an optimal sweep direction. The performance of the algorithm is limited with regards to the complexity of the environment. Observations of nonholonomic constraints were presented but not included in the algorithm scope.

In [57], the complete coverage of an environment was solved with an offline exact cell decomposition approach called *boustrophedon cell decomposition*. The boustrophedon cell decomposition is an enhancement of the trapezoidal cell decomposition that results in a smaller number of cells. Where the environment is divided with a line, two cells share a common boundary. If (complete) coverage of each cell is planned independently, overlap may occur at the common boundaries of the cells. Therefore, fewer cells will result in less overlap. As in the trapezoidal decomposition, a line is swept trough a planar polygonal environment. However, when the line intersects a vertex (of the polygonal environment), a new cell is created only if the connectivity of the line has changed. Unlike in the trapezoidal decomposition, the resulting cells are not necessarily trapezoids but (at least) monotone polygons. The adjacency of the cells is determined while decomposing the environment. The coverage planning then involves two steps: planning the visiting sequence of the cells in the adjacency graph and planning a back and forth coverage path individually for each cell. The solution was demonstrated with a simulation and on a mobile robot.

In [58], two online algorithms were proposed to solve the coverage of a planar environment. The algorithms are based on an approximate cell decomposition where the cell size is twice the size of the tool. A spanning tree of the adjacency graph of the decomposed environment is used to search the coverage path and therefore the algorithms are called *spanning tree coverage* algorithms. In both algorithms, the spanning tree is constructed incrementally online. The first algorithm assigns uniform weights on the graph edges and the resulting pattern of the coverage path

resembles a spiral. The second algorithm assigns lower weights to the graph edges that are parallel to a predetermined direction and therefore provides a back and forth coverage pattern. A square tool is used to cover the environment which is incrementally divided into square cells with equal size as the tool. Four tool size cells compose one larger square cell and the adjacency of the larger cells is defined with a graph. The algorithms follow a spanning tree of the graph while incrementally dividing the area into the tool size cells online. A large cell is defined as partially occupied if it contains at least one free tool size ell. Connections of the graph vertices depend on the layout of the adjacent cells. The solution was tested in a simulation and the memory requirements and time complexity of the algorithms were analysed.

The boustrophedon cell decomposition introduced in [57] was later generalized in [59] to solve non-polygonal regions as well. In the new adaptation called *Morse decomposition* the critical points of *Morse functions* indicate the location of the cell boundaries. The trapezoidal and the boustrophedon decomposition use a line to sweep through a planar polygonal environment, whereas in the Morse decomposition, by changing the sweep function, different coverage patterns can be obtained. The boustrophedon cell decomposition is actually a special case of the Morse decomposition where the sweep function is a line $h(x, y) = x$. As in the previous exact cell decomposition approaches, the traversal of the cells is determined as an exhaustive walk on the adjacency graph (here, a Reeb graph), and finally, computing the explicit path in each cell. A brief example of applying the Morse decomposition for coverage of three dimensional surfaces was given as well.

In [60], the Morse decomposition introduced in [59] is constructed incrementally online to solve the complete coverage of an unknown environment. To construct the decomposition online, the robot needs to detect the critical points while covering the environment and update the adjacency graph accordingly. The crucial task is to ensure the detection of all critical points. The solution was further extended in [61] where the problems of online detection of critical points and completing the adjacency graph were considered in added detail.

The applicability of *genetic algorithms* has been studied as well. A genetic algorithm is a random process based on iteratively modifying an initial population with a set of predefined functions. In [62] four basic functions and three additional functions were used to modify the previous generation of paths to obtain a new one. The paths were evaluated based on coverage, obstacle avoidance, length and absence of self-intersections. Since the modification process is essentially random it is usually unable to obtain complete coverage or a feasible path for a vehicle with nonholonomic constraints as is. The solution was demonstrated in a simulation.

In an approximate cell decomposition approach the environment is typically divided into cells that are equal shape and size as the tool. A cell is assumed covered once it is visited, and the solution is often a path with sharp turns. Such solution is not only impossible for a nonholonomic vehicle but can also be less efficient. A high resolution approximate cell decomposition was applied in [63] to obtain more freedom to generate a smooth coverage path. A contour parallel coverage behaviour with smooth paths was obtained online with a set of wall following procedures and *Bezier curve* approximation. Some improvements in time efficiency compared to

previous approaches were reported based on a simulation evaluation.

In [64], another approach was proposed to minimize the number of straight parallel paths in a back and forth coverage pattern by finding the minimum width of the polygonal environment. While such algorithms existed, their objective included reducing the computational complexity. The algorithm is based on finding the so called *antipodal vertex* of each edge of the polygon and calculating the corresponding span of the polygon between the edge and the antipodal vertex. The minimum span is the minimum width of the polygon and the parallel lines of the coverage path are placed perpendicular to the direction of the minimum width. While this solves the coverage of a convex region, a concave region is decomposed based on minimum sum of widths by an approximate algorithm. Adjacent subregions of the decomposition are combined if possible. Finally, a minimum traversal of the subregions is provided. A simulation example was given to demonstrate the overall algorithm.

A similar approach on calculating the minimum width of a convex polygon was later presented in [65] to obtain the minimum number of parallel lines in a coverage path. Finding the path with least parallel lines is based on finding the antipodal vertex pairs of the convex polygon environment and an algorithm called the *rotating calipers algorithm*. The details of the algorithm can be found in [65].

Another propositon on minimizing the number of turns in a coverage path was given in [66], where an initial exact decomposition was iteratively re-optimized. Any convex decomposition can be used to initialize the algorithm. Finding the path with least turns is based on removing and relocating the cuts in the original decomposition to minimize the altitudes of the polygonal cells. The coverage of individual cells is solved with straight parallel segments perpendicular to the minimum altitude direction. Finally, the minimum length traversal of the segments is solved as a GTSP on a graph of all the segments is all the cells combined. The results were demonstrated and evaluated in a simulation.

In [67], a simple algorithm to create back and forth coverage motion was combined with the well known A* search algorithm to solve the coverage of unknown environments. A grid representation of the covered area is constructed online. The simple back and forth algorithm is applied until reaching a point where none of the surrounding area is uncovered. The path to the next uncovered region is decided by applying a smoothing variant of the A* algorithm on the grid representation of the already covered area. The solution was demonstrated in a rectangular environment in simulations and in real life.

Recently, a new approach to online coverage of an unknown environment was proposed in [68], where the idea of an Exploratory Turing Machine is used to compute the navigation decisions online based on a dynamically built time-varying potential surface representation of the environment. The environment is divided into cells and the potential of each cell is initialized to produce a back and forth coverage motion. The potential is updated online upon detecting an obstacle. Simulation and real life experiments were provided to demonstrate the solution.

Multiple Robot Approaches

One of the early studies in cooperative coverage path planning was presented in [69]. Initial paths were generated for the polygonal environment from the configuration space boundaries and a Voronoi diagram. A contour parallel approach was chosen to avoid collisions, but the reasoning behind this choice was not given in detail. The coverage path was solved as a *Chinese Postman Problem* on a graph that connects the initial contour parallel paths. The coverage path was divided and assigned among multiple robots based on minimising the maximum travel in the resulting set of paths. The solution was demonstrated in a simulation and a real life experiment.

In [70], [71], a neural network approach was proposed to solve the complete coverage of an unknown time-varying environment with multiple cleaning robots. The algorithm also allows sudden changes in the environment. The environment is discretized into squares whose diagonal is equal to the tool radius. The activity of the neural network model represents the covered, uncovered and obstacle locations. The uncovered areas have global attraction through neural activity propagation, whereas the influence of the obstacles is only local. Other robots are treated as moving obstacles. The path is planned one neuron at a time and the next location is decided based on the neural network activity and the current location of the robot. The current location is included in the decision to avoid unnecessary turns. This approach, demonstrated in a simulation, results in a coverage pattern of back and forth parallel lines. The proposed solution can avoid both deadlock situations and collisions online.

In [72], an *integer programming* (IP) formulation was used to solve the coverage of an environment for a cooperative surveillance task performed by multiple unmanned aerial vehicles. The environment is known and it contains polygonal no-fly zones i.e. obstacles. A five constraint IP formulation was presented, including boundary conditions, collision avoidance and no-fly zones. The solution is given as a set of curvature functions, one for each vehicle. The complexity of the problem representation is reduced by discretizing the search space, time and the curvature functions. To further reduce the computational effort a receding planning horizon with a terminal cost was included in the IP formulation. The coverage behaviour is affected by the cost function, balancing between coverage and reaching a target location to finish the task. By adjusting the planning horizon, computation time can be exchanged for goodness of the solution (and vice versa). The solution was demonstrated in a simulation, however, one hour computation time did not provide complete coverage.

In [73], the boustrophedon coverage algorithm was extended to consider cooperative coverage by multiple robots while minimizing repeat coverage. Two types of communication between the robots was considered: restricted and unrestricted. In both cases the environment is unknown with both static and moving obstacles (other robots) and the cell decomposition is constructed online. Two similar approaches were proposed for the case of unrestricted communication. In the other, the environment is divided into equal sized regions and the regions are assigned among the robots. If an area is disconnected due to an obstacle, the unreachable parts are *auctioned* to other robots of the team. The other approach divides the environment into small

cells with twice the width of a robot. After completing its current cell, a robot selects the closest uncovered cell and informs the other robots of the decision. No explicit consideration of possible collisions was presented in detail.

Another type of application is an application for persistent coverage, for example, the continuous surveillance of an area of interest. The problem is often formulated as continuously covering an area where the level of coverage declines over time. For the purposes of this thesis, the concept of persistent coverage is interesting because in both cases the area is covered several times. The similarity is most evident in a collaboration of multiple robots, when an already covered area is later covered by another robot to maintain the desired level of coverage. Even if the robots have identical tasks, the situation and the challenge that follows are similar as in this thesis.

In [74], optimal paths for multiple robots were computed with regards to coverage quality. However, the environment was divided into regions and the regions were assigned among the robots and therefore, collisions were not considered. In [75], an offline cooperative solution was presented to solve the problem of persistently covering a finite set of interest points. In many applications the objective is to pass over every point of the area, whereas here, the time spent at each point of interest was considered as well. The area is covered periodically by multiple robots and all robots visit all points they are able to reach. Therefore collisions may occur. The solution is computed in three parts. The path of each robot is first solved individually as a TSP. The optimal coverage times are solved as a mixed integer linear program. Finally, to avoid any collisions, the individual periodic coverage paths are scheduled by shifting the plans to obtain a collision free combination, however, such solution may not exist. Experimental results were provided for the coverage of only a few distinct points.

In [76], the problem of multirobot persistent coverage was solved with a control law instead of planning explicit paths. The two objectives of coverage and collision avoidance were combined in a dynamic online overall control law. A weighted combination of a local and a global control law constitute the coverage control law. Collision avoidance is considered with an additional repulsive control law where the obstacles include both static obstacles as well as other robots. The overall control law is a combination of the coverage control law and the repulsive control law. The result is a dynamic, reactive online solution to persistent multi-robot coverage. Proof of safe navigation was provided. Simulation results were presented to illustrate the performance of the solution.

Another approach to persistent multi-robot coverage based on a control law was presented in [77] where an unknown environment was covered by multiple unmanned aerial vehicles. The area of interest was simplified as a two dimensional rectangular grid. The approach is distributed and requires only local communication inside a defined communication range. Each vehicle updates their own *age map* and *feasibility map*. The age map contains the information of the last coverage time of each grid cell whereas the feasibility map is used to consider the feasibility (turning constraint) of the heading to each cell. Only the position information needs to be exchanged between the vehicles inside each others communication range. The position of

other vehicles is considered when updating the age map whereas the feasibility map considers the kinematic constraints, current heading and the presence of other vehicles. A vehicles next heading is decided based on a *coverage decision map* which is a weighted combination of the age map and the feasibility map. To avoid collisions, all other vehicles inside a vehicles communication range generate a repulsive force which is included in the computation of the preferred heading. The performance of the algorithm was evaluated in simulation based on different metrics of the ages of the cells. No explicit evaluation of the collision avoidance performance was presented.

In [78], the common-task multi-robot coverage problem was (again) solved by dividing the task among the robots. Two approximation heuristics were proposed for the solution, defining optimal cooperative coverage as minimizing the length of the longest coverage path. In the other solution, an optimal tour is computed and then divided among the robots. In the other, the area is divided instead and the optimal tour is planned individually for each region. The possibility of collisions was not considered explicitly.

Another neural network approach to cooperative complete coverage was proposed in [79]. The environment is decomposed into equal size and shape square cells. The neural network of each vehicle is built upon the cell grid where each neuron corresponds to one of the grid cells. The state of a neuron is either covered, uncovered or occupied, where an occupied neuron (cell) is either an obstacle or another vehicle. The neural network activity of each vehicle is updated based on changes in the environment. Collision avoidance is therefore obtained by treating other vehicles as moving obstacles. In planning the eventual path also the current heading of the vehicle is considered to avoid inefficient paths. The solution was demonstrated in a simulation where the vehicles were modelled as points.

Cooperative coverage was studied in [80] as well, where the inspection of a three dimensional infrastructure was performed by a fleet of unmanned aerial vehicles. Collision-free cooperation was implemented with separated work areas and an online collison avoidance procedure to guarantee the maximum distance between the vehicles. Some assumptions were made of collision-free paths. The three dimensional coverage path was constructed by slicing, clustering and offset of a known model of the structure. The approach was demonstrated is multiple real life experiments.

3.4 Coverage Path Planning in Agriculture

In agriculture, the coverage task is encountered especially in the process of crop cultivation. During the cultivation cycle, many tasks are carried out on the field, including ploughing, harrowing, seeding and harvesting. The tasks are usually operated with agricultural machines such as tractors. The tractors are equipped with different *implements* or *tools* to perform the different tasks. Here, the tractors are generally referred to as *vehicles* as well.

One way of traversing the entire field is to simply decide a driving direction and then drive back and forth across the field. One contiguous path across the field is often called *a swath*, *a track* or *a driving line*. At the end of each swath, a turn has to be made to enter the next swath. The turns can be made according to a specific

turn pattern as well, like the SF-pattern shown in figure 7.

The field area is generally divided into the *mainland* and the *headland*. Sometimes it is possible to turn the vehicle outside the field boundaries but many times it is not. The field is often confined for example by a forest or a trench, and the purpose of the headland is to allocate space for making the turns. The remaining area of the field is called the mainland.

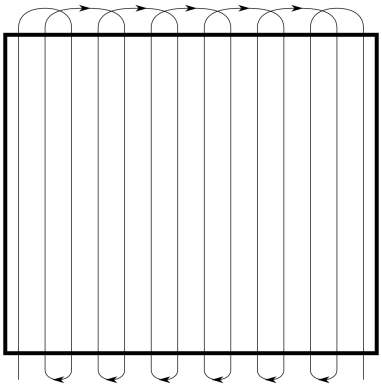


Figure 7: SF-pattern used in agricultural field operations. Adapted from [26].

In the rest of this chapter, both single vehicle and multiple vehicle approaches are reviewed. The focus and objectives of these two types of studies are often somewhat but not completely different. Some of the single vehicle approaches could be modified and extended to be applied to multiple vehicle scenarios as well. A brief review of local coordination approaches is given as well.

Single Vehicle Approaches

In [23], one of the early studies in agricultural coverage planning, a coverage planning approach was proposed to minimize the overlap between successive swaths. First, parallel swaths are generated for the mainland. The endpoints of the swaths define a set of characteristic points. A graph representation of the field work is constructed from the set of points and a set of paths defined between the points. Then, the coverage path is determined based on the search of a Hamiltonian path in the graph. Since the problem is NP-hard, a greedy search algorithm is used. A cost function considers the quality of the solution, including the path length and the number of turns, whereas an additional heuristic is used to reduce the computation time. The proposed algorithm results in a complete solution for simple and complex polygonal fields.

In [16], an automated harvesting system was described and demonstrated, including harvesting 40 hectares of crop in a continuous run. The system is capable of planning and executing a harvesting operation including a coverage plan. The coverage plan minimizes the number of passes to process the field, however, the exact details of the solution are not provided. GPS data points are obtained from surveying the field in advance. The coverage plan is designed according to the data

points and the machine measurements such that every turn is made to the successive row. Both GPS based and vision based positioning was demonstrated, while GPS proved to be the more accurate and robust choice.

In [24], the traversal sequence of the swaths was optimised to minimise the total non-working distance on the field. The computation of the optimal sequence is based on finding the shortest path in a weighted graph. Each edge of the graph is associated with a cost that corresponds to the travel distance along the edge. The non-working distance is defined as the sum of the total turning distance at the headlands and the distance of travel between the first and last track and the initial and final positions. The field efficiency is maximized with the shortest path that visits every vertex of the graph exactly once, starting and finishing at the initial and final positions. The solution was obtained by solving a formulation of the optimization task as a binary integer programming problem. The optimized sequence was able to reduce the total non-working distance by up to 50% during demonstrations. The benefits of using algorithmically optimized field patterns (later introduced as B-patterns) were further studied in [26].

In [27], two different approaches were used to solve the coverage of agricultural fields. In the first approach, in addition to using a trapezoidal decomposition, a search for the best driving direction was developed. The efficiency of a region in the decomposition is defined according to the area of the region, the length of the route and the efficiency of driving. Each step of the algorithm, the remaining area of the field is split into trapezoids, the trapezoids are merged into larger regions, the best driving direction is searched, and finally the most efficient region is chosen and removed from the field. The algorithm is repeated on the remaining area until the entire field is split. However, to enable other than straight driving lines, a more complex online approach was developed. The second approach, inspired from *model predictive control* (MPC), uses a search horizon to limit the search space for the best possible route. With the limitation, admissible computation time is obtained. Selecting the best route is based on time efficiency: the cost function is defined as the ratio of the working distance to the total driving time. To search for the best route, a simulation is made at the end of each swath. Only the first swath of the best route is applied, and then the process is repeated. The algorithm provides feasible solutions also for complex fields.

A combination of two earlier independent works was presented in [81], where dividing the field and choosing the driving direction in each region was implemented as in [27], and optimising the track sequence was implemented as in [24]. That is, an iterative split and merge approach based on trapezoidal decomposition was combined with swath sequence optimization based on a cost function minimizing the non-working distance of the machines. Also the visiting sequence of the regions was optimized, resulting in a complete plan for processing a complex field of several subregions.

The problem of optimising the swath orientation was studied in [28] as well, while additionally considering the effect of swath orientation on the field margins and coverage efficiency. A cost function was used for economic evaluation of the swath and margin configuration. The cost function aims to balance between qualifying

for field margin subsidies, loss of income from unprocessed area and the cost of an additional swath. An exhaustive search is performed over a discrete set of orientations of the swaths, including shifting the position of the swath pattern. However, only straight swaths and fields with relatively simple geometry were considered.

In [29], a custom cost function was applied to find the optimal solution of decomposition and subregion driving direction for the coverage of agricultural fields. The cost of several different headland turns was analysed and considered in the cost function. The search for an optimal decomposition and coverage direction was formulated as a depth first search in an undirected graph. The premise behind the cost function is that the coverage cost is primarily determined by the headland turns. The cost depends on both the amount and type of the turns. A recursive algorithm was developed to perform an exhaustive search for the minimum cost solution over all possible decompositions of the field based on the graph representation. First, the optimal driving direction is determined for the entire field, and the coverage cost is later used as a threshold. Then, all possible ways to divide the field into two regions are found. The algorithm is repeated, for both regions of all the candidate decompositions, until no more valid divisions can be made. If any of the solutions provide a lower overall cost than the threshold, the solution with the lowest cost is selected. Whereas the turning cost of several types of headland turns is analysed in depth, the algorithm only allows turning to the successive swath and the optimal path between subregions is not provided.

In [21], a coverage path planning approach for rolling terrain fields was developed. The field was decomposed into subregions based on terrain features, and a so called *seed curve* was searched for each region. The coverage path was generated by offsetting the seed curve to fill the region. A cost function was used to determine the minimum cost path: in addition to headland turning cost as in [29], soil erosion and the additional cost of a curved path were considered. Heuristics were used to reduce the search space i.e. to reduce the computational time.

In [30], a genetic algorithm was used to optimise the visiting sequence of the subregions of a field. First, the mainland swaths were generated as parallel to a driving direction decided by the user. The swaths are clustered into contiguous subregions based on swath intersections with obstacle areas on the field: the number of the regions depends on the previously chosen swath direction. Next, the headland path is obtained, followed by the optimization of the visiting sequence of the subregions. Depending on the amount of regions, a genetic algorithm is used to find both the optimal visiting sequence and the optimal entrance point of each region. Even if the swath orientation was not optimized, such an algorithm could be used instead of a user-defined driving angle.

In [82], existing methods were combined to solve the coverage problem while optimising the orientation and the sequence of the swaths and the choise of the headland turns. The optimization is based on minimising the non-working time with respect to both turning and servicing. A headland is generated for turning and the mainland is filled with straight parallel swaths. A coverage plan is determined for each swath orientation over 180 degrees with 1 degree intervals. Two types of headland turns presented previously in [24] are considered. Each pair of swaths is

connected with either of the two types of turns depending on the distance between the swath endpoints. Travelling all the swaths is solved as a TSP with a heuristic algorithm. Finally, the swath sequence may be altered to optimize the usage of the vehicle capacity. A service stop is required when the vehicle capacity is not enough to complete the next swath. Scenarios involving neutral, input and output material flow were presented as demonstrations. It was observed that the optimal swath orientation is not necessarily the same for minimum turning time and minimum servicing time.

Another approach on minimizing the non-working distance during field coverage was presented in [32], where a new swath traversal pattern was combined with an optimal decomposition of a complex field. The proposed approach includes optimization of both the visiting sequence of the subregions and the traversal sequence of the swaths. The vehicle characteristics were considered by modelling it as a Dubins vehicle. The optimization problem was modelled as a Generalized Travelling Salesman Problem (GTSP) that can be solved using a variety of existing algorithms. The GTSP is defined in a directed graph where each vertex is a directed path across one of the swaths. There are two vertices for each swath: one in both directions. The two vertices of each swath form a cluster resulting in m clusters if there are $2m$ vertices. The vertices in the same cluster are not connected since each swath should be travelled only once. A cost matrix is constructed such that the cost between any other pair of vertices is based on their (Dubin) distance: the solution minimizes the total non-working distance while allowing the exit and entrance points of successive swaths to be located on different sides of the field. The algorithm can be extended to minimize the total non-working distance for a complex field. The swaths of all subregions are connected and considered in the cost matrix as if they were in the same field. If some connections between subregions are forbidden, the cost between any of their vertices is set to be a large positive number. With this approach, it is not necessary to complete a subregion before moving to another one. Improvements were reported when compared to previous solutions, however, the assessment was conducted only in simulation and with relatively simple fields.

The challenges caused by variation of slope were further studied in [31], and a new approach was developed that ensures full coverage regardless of the field topology. An approach to evaluate the efficiency of coverage algorithms for varying elevation terrain was developed, and the efficiency of the developed coverage approach was demonstrated in both simulation and real-life.

Multiple Vehicle Approaches

Approaches involving cooperation of several vehicles have been presented as well. In a typical scenario, either a fleet of primary units and support units are cooperating, or multiple vehicles perform the same task and somehow the field is divided and allocated among the units of the fleet.

In [83], several types of agricultural field operations involving multiple vehicles were characterized and then formalized as different variations of the VRP or the TSP. Single and multiple primary units were considered in neutral, input and output material flow operations. The field work is represented as a graph where each swath

is defined with two vertices. The corresponding edge is given a zero cost to enforce the solution to travel each swath. In addition, a demand is assigned for every vertex: depending on the operation, the demand can be either deterministic, stochastic or unknown. A probabilistic variation of the classic VRP can be used for operations involving uncertainties. For a neutral material flow operation the demand is defined as zero. The work was continued in [84] to consider single or multiple support units in the operations. Another study that involves cooperation of primary and support units during a coverage task was presented in [34], however, the focus is on path planning for the support unit.

In [85], the problem of complete field coverage was considered as a multiple vehicle routing problem as well. A mixed-integer linear programming approach was used to formulate and solve the problem. The length and orientation of the swaths is assumed to be known, whereas the travel direction of the swaths is a part of the optimization. There are four options of travelling from one swath to another: each swath can be travelled in two directions and the two swaths can be travelled either in the same direction or opposite directions. Two different formulations of the mixed-integer linear program were provided: one that minimizes the total travel distance and one that balances the workload among the vehicles. Again, since the travel distance along the swaths is fixed, minimizing the travel distance is reduced to minimizing the non-working distance between successive swaths. The other formulation balances the workload among the vehicles and therefore the travel distance along the swaths needs to be considered as well. The details of the mixed-integer linear programs as well as the two alternative objectives can be found in the paper.

In [35], Simulated Annealing and a cost function with multiple optimization criteria were used to determine the optimal track sequence including refilling for a fleet of heterogeneous vehicles. The cost function to assess the goodness of a solution considers (1) the sum of travelled distances, (2) the sum of input costs such as fuel and (3) the maximum of time consumed by each vehicle. To minimize all of the criteria at once, a linear combination is used. The problem was formulated as a combinatorial optimization problem to allocate the tracks among the vehicles and to optimize the track sequence. The solution is expressed as two vectors: one containing the track sequence and allocation, and the other containing the refill schedule relative to the completion of the tracks. The solution itself eliminates the possibility of collisions in the mainland, however, a solution to avoid collisions in the headland was not described in detail. The solution was demonstrated in simulation with simple rectangular fields. The tests were performed with different set-ups, where the amount of vehicles and swaths were varied. The units of the fleet have different tank capacities, working speeds and turning radii. Also the fleet characteristics were observed to influence the solution. To improve the optimization process a new operator for the Simulated Annealing method was proposed in [86].

A centralized planning and control system for a fleet of small agricultural vehicles performing a seeding task was presented in [87]. For positioning, each vehicle and the central control unit is equipped with an RTK GNSS system. Planning, task assignment, optimization and supervision are all handled by the central unit. An initial plan is designed based on user input such as the field boundary and the number

of vehicles. First, a headland is generated and an entrance and exit location to and from the field is defined. A fixed driving direction is defined for the swaths, and the swaths are driven back-and-forth. For a complex field, subregions are generated where the swaths intersect the inner or outer boundaries of the field. As the vehicles are identical, the workload is distributed as evenly as possible to guarantee maximum efficiency regarding time (either the same number of lanes or the same distance to drive). The refills required during the task are planned individually for each vehicle. The swaths are assigned to the vehicles so that collisions in the mainland are not possible, however, to prevent collisions in the headland (to and from refill) the central unit must supervise and coordinate the vehicles. The approach has been demonstrated in a simulation where the interface between the control unit and the simulation is identical to the interface to the real vehicles. In simulation, the system has been shown to be fully functional with a fleet of vehicles.

In [88], reducing the field completion times with a fleet of vehicles was formulated and solved as a Vehicle Routing Problem. The VRP was defined in an undirected graph, where three vertices were defined for each swath: one for both endpoints and one for the midpoint. In the graph, the midpoint is connected only to the endpoints of the corresponding swath. A cost was assigned to every edge in the graph. For an edge between a swath endpoint and a corresponding midpoint, the cost is half the travel time of the entire swath. For an edge between two swath endpoints in the same headland, the cost is the turning time from the first swath to the other. All other edges in the graph are considered invalid. A large positive value was given to any invalid connection in the cost matrix. The reason for defining a midpoint for each swath is twofold: while enforcing the solution to travel every swath, it also allows assigning a non-zero cost for each swath. If only non-working distance is considered as in [83], a swath can be defined as one edge with a zero cost. Finally, a cost function was developed to consider both the sum of working times for all vehicles and the time for the last vehicle to complete its task. The solution to the VRP is a set of routes including a route for each vehicle. No vertex is visited twice i.e. every vertex is contained in only one of the routes. Two existing algorithms were applied to solve the VRP: one heuristic algorithm and one meta-heuristic algorithm. The heuristic algorithm was modified to obtain a solution to the multiple vehicle problem. Whereas the heuristic algorithm produces feasible solutions very fast, the meta-heuristic algorithm took hours to produce the solutions. However, increasing the number of vehicles resulted in notable reductions in the field completion times with the meta-heuristic algorithm. For initial testing, the vehicles were considered identical with zero mass and holonomic steering. A heterogenous fleet could be considered with individual cost matrices.

Local Coordination Approaches

It appears that the problems of multi-vehicle coverage and collision avoidance are often considered two separate issues or at least solved as such. To expand on the issue of collision avoidance some approaches focused on local coordination are presented.

In [33], a method for developing a two-unit platooning system for agricultural

vehicles was presented. The system enables a driverless tractor to follow a leading tractor at a given lateral and longitudinal offset. RTK GPS equipment was used for positioning of the vehicles. The position of the leading vehicle is obtained at regular intervals. The position for the following vehicle is calculated using the given offset and curve fitting is used to obtain its path. During turning in the headlands the turning paths of the vehicles intersect. Therefore, the following vehicle is halted while the leading vehicle turns. To further enhance the safety of the system and to prevent any collisions, there is a virtual safety zone around the following vehicle. In case of violation of the safety zone, the operation is halted by a real-time program. Only preliminary simulation tests were conducted but not reported in detail.

In [89], the issue of avoiding collisions in multiple vehicle operations was addressed with a collective tracking controller based on model predictive control. The problem of neither coverage nor path planning is considered: a global plan needs to be provided by a higher level planner. The purpose of the tracking controller is to solve local conflicts that may have been disregarded or were not possible to anticipate in the global plan. Two modes of operation are considered: in a group of vehicles, (1) one vehicle acts as a master to one or more other vehicles, or (2) all vehicles are independent and perform their individual tasks. A hierarchy of operation modes is also possible, however, a vehicle can only have one master. Each vehicle of the fleet has its own nonlinear model predictive tracking controller that computes the steering and velocity commands. To solve the control action for the next time step the basic idea is similar as in model predictive control. At every time step, the optimal control problem is solved over a finite predicting horizon. However, only the control action for the next time step is applied while the rest of the control sequence is discarded. At each following time step, the optimization problem is solved again over the prediction horizon, until the goal state has been reached.

The vehicles motion is described by a discrete state equation which is a function of the previous state and control inputs. A cost function is used to optimise the control sequence over the prediction horizon. Three terms are penalised: (1) the total predicted tracking error, (2) the predicted deviation from the last state and (3) the control effort over the prediction horizon. The tracking error is the difference between the desired state as computed by the global planner and the result of the optimal control action as computed by the tracking controller. The control optimization problem is to minimize the cost function over the prediction horizon. To extend the controller to consider local coordination among several vehicles, an additional penalty term is introduced to the cost function. The additional term (4) considers the minimum Euclidean distances to all nearby vehicles over the prediction horizon: it penalises the proximity of other vehicles as if they were moving obstacles. The collective approach requires that the prediction horizon for all the vehicles is the same: each vehicle is required to broadcast its motion trajectory to all other vehicles in its proximity. The trajectory includes both the current state and the plan over the prediction horizon. The optimization problem can be solved with existing methods. The emergent behaviour can be tuned with common and vehicle specific parameters involved in the equations. The parameters determine the balance between tracking error and safety clearance during conflicts. Individual parameters include state

and control input restrictions as well as the *rank* of the vehicle which indicates its precedence among the fleet. The prediction horizon is included in the common parameters. [89]

A simulation study was conducted as a proof of concept for collision avoidance and formation control in different scenarios. In the simulation, the tracking controller was demonstrated to avoid both head-on collisions and collisions resulting from intersecting trajectories. Given a control input of steering and speed, the vehicles are able to coordinate either by altering their speeds or by deviating from their paths (or both). The parameter values used in the simulations are reported in the paper. A sampling period of 0.1 seconds was chosen for the controller together with a prediction horizon of 65 steps, resulting in a prediction of 6.5 seconds in the future at each time step of the controller. [89]

3.5 Summary

The available coverage path planning research is plenty in both agriculture and various other application areas. Therefore this presentation of earlier work is not exhaustive but demonstrative. Several different application related aspects of coverage path planning were identified in the literature. A brief summary of these application key points is outlined in table 1.

Table 1: Application related key points in coverage path planning.

Environment	
known	unknown
static	dynamic
determinate	uncertain
simple	complex
2D	3D
inside	outside
Vehicles	
holonomic	non-holonomic
single	multiple
homogenous	heterogenous
Tasks	
one	multiple
common	specialized
independent	coupled
preallocated	not allocated
Planning	
centralized	decentraliced
optimal	heuristic
offline	online
replanning	no replanning
Coverage	
static	dynamic
complete	approximate
one-time	persistent

To obtain complete coverage for a complex field, an exact cellular decomposition is often used. The coverage path for each simple region is typically a set of straight parallel swaths that are processed in alternate directions i.e. back-and-forth. Other than straight swaths have been considered as well. Optimising the swath sequence as well as the direction of the swaths has been proposed in several studies. Optimization is often based on a cost function that measures efficiency in terms of non-working distance or time. Many studies acknowledge the number of turns (and service stops) as the main factor of the coverage cost. The problem of finding an optimal traversal sequence is often formulated as a variant of the Travelling Salesman Problem or the Capacitated Vehicle Routing Problem, and a graph is used to represent the field and the swaths. With a fleet of vehicles, the mainland is often divided so that each vehicle operates on its own region. To avoid collisions in the headland or in a shared workspace, a separate supervisory controller is often used. Collision avoidance is not always described in detail. Many of the algorithms are primarily offline, however, also feasible online algorithms have been demonstrated.

Many studies consider different types of cooperative scenarios where multiple vehicles operate simultaneously on the same field. However, a thorough search of the relevant literature yielded no related work where two vehicles perform sequentially

dependent individual coverage tasks simultaneously on the same field, sharing the same workspace. The relevant cooperation scenarios can be roughly divided into the following three cases:

(1) A primary unit is collaborating with a support unit. Both units are needed to complete a common task. The primary unit cannot continue its work until the supporting unit has performed its task. For example, harvesting and unloading.

(2) A task is divided among several independent primary units. Several units provide redundancy: all units are not necessarily needed to complete the overall task. Field completion times can be reduced compared to using a single vehicle.

(3) Several primary units are performing sequentially dependent tasks. Having two units performing two tasks, the second unit is not needed to complete the first task. The second unit can only work on areas that the first unit has already processed.

Cases (1) and (3) share the following similarity: in the case of harvesting and unloading, the support unit is not allowed on the unharvested part of the field, and therefore, its free workspace is restricted by the progress of the primary units task. However, the support unit is not required to perform a coverage task. Cases (2) and (3) are similar only in that all units are performing a coverage task. In both cases, adjacent swaths and headland turns are main concern for spatial conflicts.

The objective of this thesis is to provide a solution to case (3). At least three concept level approaches can be found to implement a global mission planner for case (3) including collision avoidance:

(1) The vehicles are assigned strictly separated workspaces to avoid collisions. Both headland collisions and mainland collisions are avoided with separated workspaces. Coverage planning for both vehicles can be done offline. The adaptability of this type of algorithm is questionable.

(2) All units share the same headland. In the mainland, the second unit is allowed only on the areas that have been processed by the first unit. Collision conflicts in the headland are solved by an independent local coordination controller once they are encountered. Coverage planning for both vehicles can be done offline.

(3) All units share the same headland. In the mainland, the second unit is allowed only on the areas that have been processed by the first unit. The local coordination is incorporated in the global path planner as the coverage is planned online for all vehicles involved. Coverage planning for both vehicles can be done offline if the route is planned as trajectories but failure in reaching a waypoint in time might result in a failed mission.

4 Mission Description and Definitions

In this thesis, the word *mission* is used to describe an overall objective to complete one or several related tasks. Thus, a mission is considered complete when all of the mission tasks are finished. In this context, a *task* is defined as one work phase to be performed by a field machine on the field.

The mission considered in this thesis includes two tasks related to the sowing of a field. The first task is to prepare the field for sowing (with a disc cultivator), and the second task is sowing. A simplified illustration of the real-life task set-up is shown in figure 8. This illustration highlights the most important components of the high level architecture of the system. A short description of the components is given in table 2. The numbering in figure 8 corresponds to the numbering in table 2.

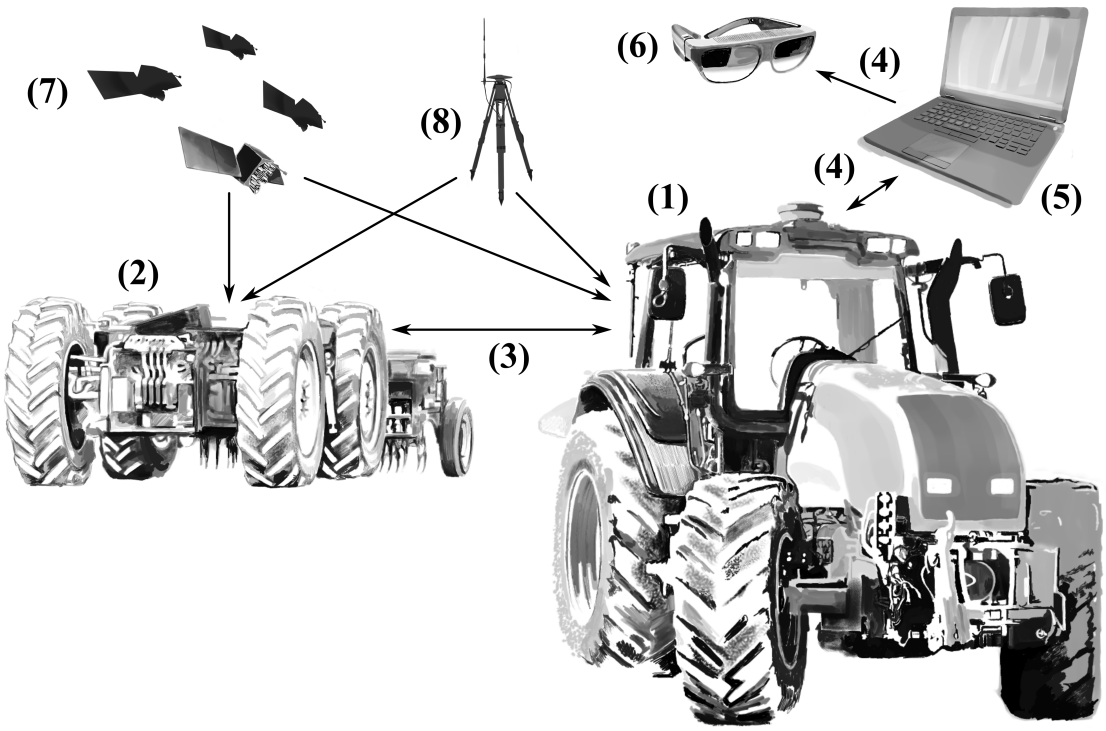


Figure 8: The high level system architecture and connections.

Table 2: Description of the high level system architecture in figure 8.

	Description
(1)	The semi-autonomous tractor performs the 1st task
(2)	The fully autonomous driverless tractor performs the 2nd task
(3)	Radio link between the tractors
(4)	UDP communication
(5)	Mission Planner computer
(6)	Augmented reality smartglasses
(7)	GPS satellite constellation
(8)	RTK GPS base station

The focus of this thesis is cooperative coverage path planning with sequential tasks. The overall task or *mission* is to process an agricultural field twice: first with a disk cultivator and then (immediately after) with a seeder. The tasks are to be performed autonomously and simultaneously by two different vehicles. The following statements give a general definition of the mission:

(1) Two different operations (tasks) are to be performed on an agricultural field

- cultivate the field area (1st task)
- sow the field area (2nd task)

(2) The tasks require two vehicle cooperation

- one semi-autonomous tractor (with human operator)
- one fully autonomous tractor (driverless)

(3) The vehicles are operating on the field simultaneously

(4) The tasks are preallocated

- the semi-autonomous tractor performs the 1st task (cultivating)
- the fully autonomous tractor performs the 2nd task (sowing)

(5) The tasks are strictly sequential

- 1st task is independent (can be performed on any unprocessed area)
- 2nd task is dependent (can be performed on areas where the first task has already been performed)

(6) The vehicles are equipped with hitch mounted implements

- the semi-autonomous tractor is equipped with a disk cultivator
- the fully autonomous tractor is equipped with a seeder

(7) The field has been previously ploughed

The solution will be applied to a real-life test set-up and therefore certain assumptions and technical requirements are to be considered. The technical requirements for the solution and the real-life conditions that should be considered in its development include

(1) A mission planner software is developed to operate the overall mission

(2) Route planning is centralized

- the routes for both vehicles are computed in the same process running on one computer

(3) The interface between the Mission Planner and the vehicles is implemented over UDP

- Guidance waypoints sent to the vehicles
- State feedback from the vehicles to the Mission Planner

(4) The position of the vehicles is known with the accuracy and precision of RTK GPS

- both vehicles are equipped with RTK GPS capable receivers

(5) Communication at approximately 100 ms intervals during normal operation

(6) Communication is unrestricted

- UPD messages are received approximately on 100 ms intervals
- A radio link between the vehicles

(7) KKJ3/YKJ projection is used for location coordinates

- Transformation from and to GPS WGS84 coordinates is performed in the vehicles

(8) Possible real-life condition failures include (but are not limited to)

- errors or loss of RTK GPS signal
- network issues like UDP packet loss or latency
- failure of the radio link between the vehicles

(9) Both vehicles navigate based on the GPS receiver location (placement on the vehicle)

As for the algorithm itself, some assumptions were made regarding the representation of the environment, the vehicles and the implements. In addition, certain algorithms that were required for the solution were not implemented as a part of this thesis but were assumed to be provided. The algorithm was developed given the following assumptions:

(1) The field is defined with multiple polygons where

- one outer polygon describes the field boundary and
- one or more inner polygons describe stationary obstacles

(2) The environment is known including

- the area boundary
- the location of stationary obstacles
- the location of both vehicles (approximately every 100 ms)

(3) A grid representation is used to track the progress of the tasks but the vehicles are guided with exact GPS coordinates

(4) A separate headland algorithm is required because of application related constraints

(5) A turn path algorithm is provided (for turns between successive swaths)

(6) Vehicle related constraints need to be considered, for example, the curvature of the path is restricted

(7) The implements can be modelled as lines with constant widths

(8) The paths are planned for the center points of the implements and then transformed to the GPS receiver location

(9) Anything beyond the UDP interface is not developed as a part of this thesis

5 Implementation of the Mission Planner Approach

In this thesis, two different algorithms were implemented to solve the cooperative coverage path planning problem with sequentially dependent tasks. Both algorithms are based on the idea of computing short simultaneous paths that do not overlap, and then scheduling them in real-time.

The objective of this thesis is to provide an algorithm that can be used in a real-life environment with real tractors and equipment. In addition to the real-life tests, a simulation environment and evaluation was originally planned as a part of this thesis. Therefore, the Matlab class called *Mission Planner* that includes the cooperative coverage path planning algorithms was designed to be compatible with two different simulations and a C# solution called *Mission Operator*. A brief description of the development and testing architecture is given in chapter 5.1. Eventually, the simulation environment could not be used to test the algorithms due to technical issues and time constraints.

The solution to the cooperative coverage path planning problem is two-part: it consists of a set of utility functions and a class implementation that contains the path planning algorithms, both implemented in Matlab R2018a. The set of utility functions, described in detail in chapter 5.2, was developed for conflict detection and to solve the general coverage path planning problem. The algorithm implementation in the Mission Planner class is described in detail in chapter 5.3. A centralized approach that is run on one computer was chosen for the implementation. The Mission Planner solves the paths for the vehicles but does not schedule them. The purpose of the Mission Operator is to operate the Mission Planner algorithms during the real-life test drives. The real-time scheduling is done in the Mission Operator. The Mission Operator is described in detail in chapter 5.4.

5.1 Development Architecture

The Mission Planner is a Matlab class where the coverage path planning algorithms are implemented as class methods. The Mission Planner was developed to be compatible with the Mission Operator C# solution and two different simulations. To be used in the C# solution, the Mission Planner Matlab code is assembled into a dynamic-link library (DLL) and included as a reference in the solution.

A non-real-time simulation was designed for initial testing and validation of the Mission Planner algorithms in Matlab and Simulink. To test the full system architecture including the UDP interface, a type of a Hardware in the Loop (HIL) simulation was designed. In the real-life tests the Mission Operator sends the commands to the vehicles over UDP. The interface to the real-time HIL simulator was designed to be identical to the interface to the vehicles. The non-real-time simulation, on the other hand, was planned to provide a fast validation of the algorithm for a fast development cycle. The Mission Planner can be included in the non-real-time simulation with an extrinsic wrapper that holds a persistent instance of the Mission

A simplification of the development and testing architecture plan including the

interfaces between the modules is shown in figure 9. A visualization for the simulations was also developed for visual validation of the algorithms. The visualization can be included in both simulations.

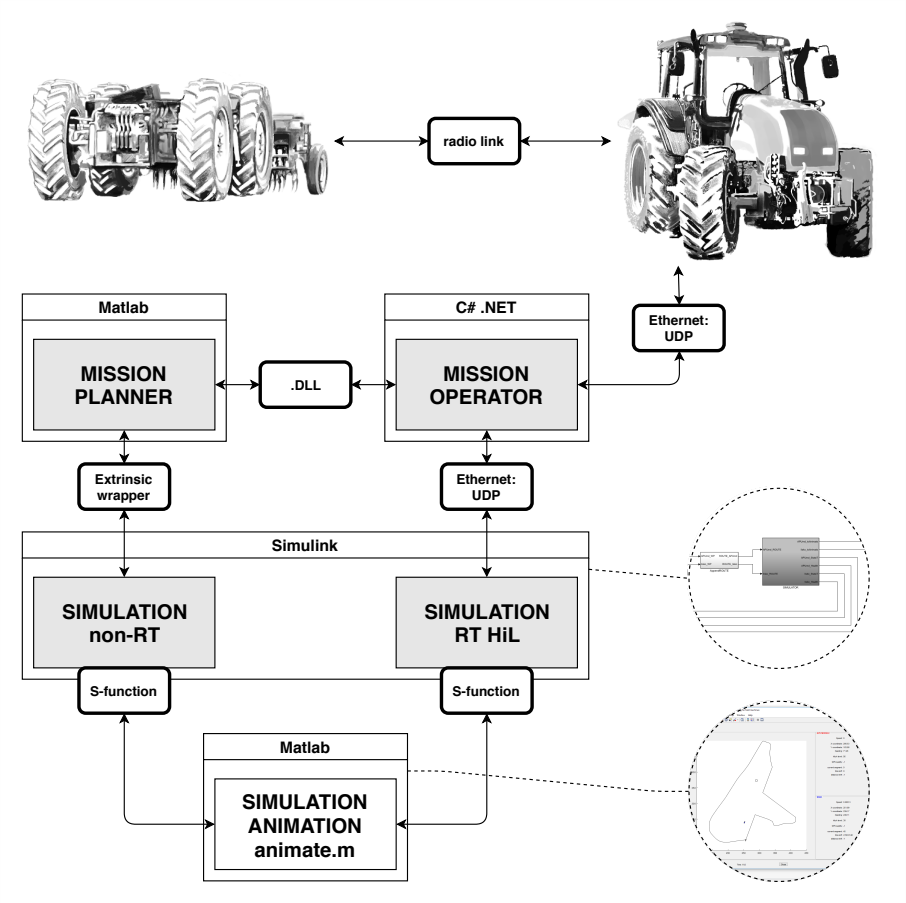


Figure 9: Development and testing architecture.

5.2 Utility Functions

A set of utility functions was developed in Matlab R2018a for conflict detection and to solve the general coverage path planning problem. Solving the cooperative coverage path planning problem for any polygonal area consists of solving several subproblems, including

(1) Field decomposition

(2) Headland swaths

(3) Mainland swaths

(4) Turn path calculation

(5) Solving the swath order

(6) Start position and first swath

(7) Coverage map update

(8) Available swaths for the second vehicle

(9) Conflict detection

(10) Consideration of real-life constraints

For both vehicles, the swaths for both headland and mainland are solved in advance, and only the order in which the vehicles drive these swaths is calculated online. The second vehicle is allowed to drive only those swaths that cover an area that the first vehicle has already completed. Continuous update of the progress of both tasks is needed to update the available swaths for the second vehicle and to monitor the task status. The next swath should be chosen for both vehicles before the vehicles reach the end of the previous swath. To proceed from one swath to another, a turn path needs to be calculated. A simple method of collision detection is used to ensure that the simultaneous paths for the two vehicles are not in conflict. Finally, the cooperation is implemented by scheduling two non-conflicting paths, one path for each vehicle, at a time.

Field Decomposition

A field decomposition for concave field areas was not implemented as a part of this study. Therefore the solution is able to solve the coverage path consistently and reliably only for convex fields.

Generating the Headland Swaths

For the algorithms developed in this thesis, a headland was created everywhere around the field area. In many cases it is not necessary, but it does provide some flexibility for turning and therefore for determining the mainland swaths afterwards. This approach also guarantees that the corners of the field can be operated also when the turning area is limited to the field.

Given a polygon boundary of the field, the following algorithm for generating the headland swaths was developed. A polygon *offsetting* algorithm was used to generate the outlines of the headland swaths. The field boundary is simply *inset* i.e. *offset inwards* by a distance derived from the tool width. To obtain four nested rounds of swaths in the headland, the insetting is performed four times. The polygons are later referred to as headland swath polygons. An open source freeware library from [90] was used for the offsetting.

The final headland swaths are determined from the headland swath polygons. In the algorithm the vertices of the headland swath polygons that outline the headland swaths are processed clockwise. Two line segments are connected by each vertex of a polygon: in the processing order of the vertices, they are called the *previous* one and the *following* one. This notation of precedence also applies to the vertices.

For an arbitrarily shaped polygonal field, some vertices of the headland swath polygons can be too steep to drive along the edge without lifting the implement and making a turn at the vertex. Such a vertex is referred to as a *critical vertex*. The criticality of a vertex is determined based on the direction change between a previous and a following line segment as in [27]. A circle with a radius of twice the turning radius of the vehicle is centred on the vertex: the direction change is calculated from the line segments that intersect the circle. If the direction change is more than 30 degrees, the vertex is a critical vertex.

The headland area is limited by an *outer* and an *inner* boundary. The outer boundary is equal to the field boundary, and the inner boundary is determined by the number of headland swath polygons and the tool width. The line segments of the outer and the inner boundary connected at the critical vertices are used to determine the final headland swath vertices between which a turn has to be made.

Each vertex in the headland polygons that correspond to a critical vertex in the inner boundary is converted into two new vertices: between these vertices, a turn has to be made. The two vertices are later referred to as a turning point. A separate turn path algorithm is used to generate the actual turning path between the vertices, and in fact, the turn can be made to another swath than the next headland swath as well.

In every polygon including the outer headland boundary, the inner headland boundary and all of the headland swath polygons, the number of vertices is the same. This does require the assumption that during the insetting, the number of vertices does not decrease, which is indeed possible in certain cases. For each vertex in each polygon there is a corresponding vertex in every other polygon: in addition to having the same amount of vertices, the vertex indexes in all polygons are the same.

A simplified step-by-step description of the headland swath generating algorithm is given in the following. The steps of the algorithm are shown for a simple non-convex field in figure 10.

(1) Obtain the *outer* boundary polygon of the headland. This is equal to the boundary of the field area that has to be processed during the task. The outer boundary is shown in figure 10a.

(2) Create the outlines of the headland swaths by insetting the *outer* boundary polygon. The first headland swath polygon is inset by one half of the implement width from the outer boundary, whereas the following headland swath polygons are inset by one implement width from the previous one. In the example in figure 10b, there are four headland polygons.

(3) Determine the *inner* boundary polygon of the headland by insetting the last headland swath polygon by one half of the implement width. The inner boundary is shown in figure 10c.

(4) Determine all the critical vertices of the *inner* boundary polygon. A critical vertex is a vertex where a turn has to be made. How to determine the criticality of a vertex has been described above. All the vertices of the example field are critical, as shown in figure 10e.

(5) To determine the turning point vertices for each headland swath polygon, for each critical vertex in the inner boundary:

   (a) Determine if the headland swath polygon is *convex* or *concave* at the critical vertex i.e. whether the internal angle between the two line segments connected by the critical vertex is less or more than 180 degrees, respectively. In figure 10f, the angles β and γ indicate a convex vertex, whereas the angle α indicates a concave vertex.

   (b) For a *convex* vertex, perform the following, as illustrated in figure 10g:

      i. From the *inner* boundary polygon, take the line segment between the critical vertex and the previous vertex.

      ii. Extrapolate the line segment from the critical vertex to intersect all the headland swath polygons.

      iii. Cut the headland swath polygons at the intersections: the intersection points are new vertexes.

      iv. Remove the new line segments between the vertices corresponding to the critical vertex and the intersection points.

      v. Extend the previous line segments of the headland swath polygons to intersect the *outer* boundary polygon.

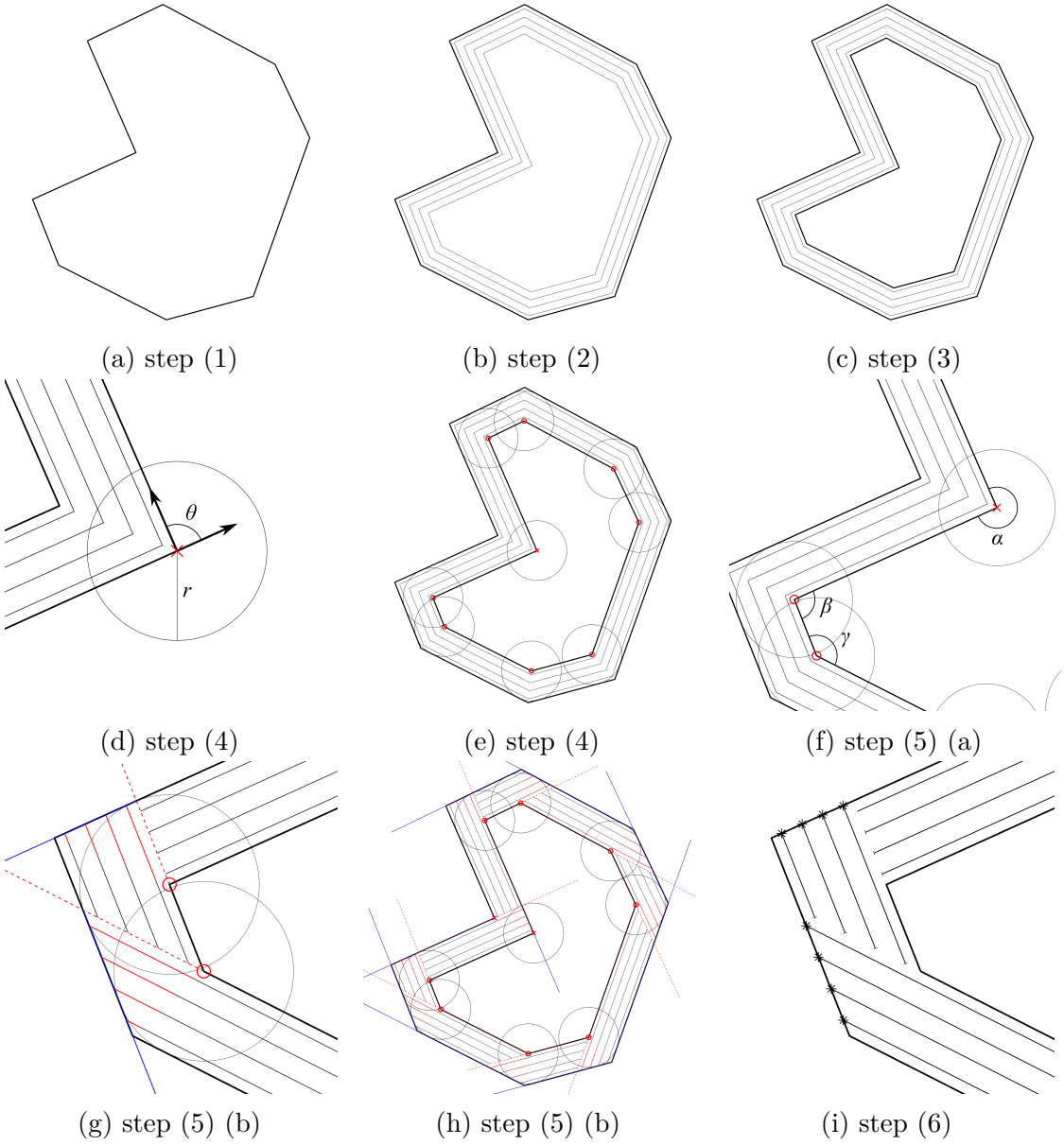
      vi. Cut the extended line segments at the intersections: the intersection points are new vertexes.

      vii. The two new vertices are a turning point: between them, a turn has to be made.

   (c) For a *concave* vertex, perform the same steps as for a *convex* vertex, but so that the roles of the *inner* and the *outer* boundary are swapped.

(6) Finally, the headland swaths can be driven only in one direction to guarantee that the corners of the field can be operated without crossing the headland boundaries. The first vertex of each headland swath should be located on the inner or the outer headland boundary depending on whether the corresponding critical vertex is convex or concave. In figure 10i, an asterisk shows the first vertex of the swaths.

The headland swaths are generated such that the headland can be processed even when the tractors need to always stay inside the field boundaries. This also requires that the headland swaths can only be driven in one direction beginning from the endpoint that is located on the outer or inner boundary of the headland. With the given algorithm, the resulting swaths in the headland are such that the headland should be driven counter clockwise. This can be seen e.g. in figure 10i, where it is evident that otherwise the vehicle should drive outside the field boundary to process the entire swath.



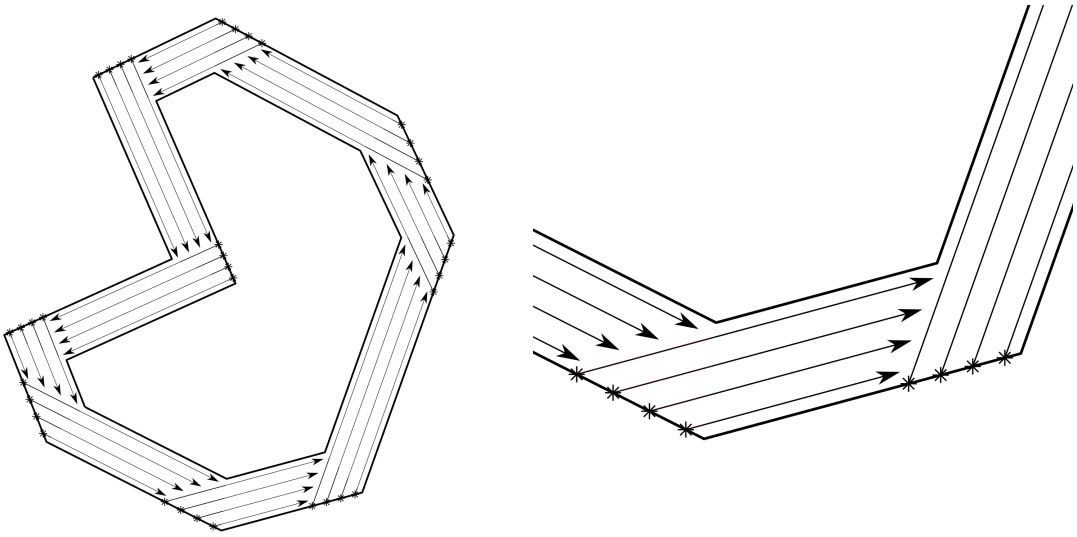
(a) step (1) (b) step (2) (c) step (3)

(d) step (4) (e) step (4) (f) step (5) (a)

(g) step (5) (b) (h) step (5) (b) (i) step (6)

Figure 10: A simplified description of generating the headland swaths.

The end result i.e. all the headland swaths are shown in figure 11a. The first vertex of the swaths are shown with an asterisk and the direction of the swaths is shown with an arrow. Notice that the swaths always begin at the outer boundary or at the inner boundary of the headland depending on whether the corresponding critical vertex was convex or concave. The direction of the headland swaths is shown in detail in figure 11b. When the swaths are driven in this direction (and if the headland is wide enough) it is possible for the vehicles to stay inside the headland boundaries when driving any of the headland swaths.



(a) All headland swaths. (b) Driving direction.

Figure 11: The resulting headland swaths from the headland swath algorithm.

The order in which the swaths are eventually driven is computed later during the the cooperative algorithm, where the order is optimized based on an efficiency function. These swath centrelines provide a geometric solution to how to cover the headland area. These centrelines are later modified during the execution of the cooperative algorithm to consider for example the placement of the GPS receiver on the vehicle. The vertices and the line segments are converted into waypoints where also other parameters such as speed are assigned.

Lastly, it should be observed that depending on the field shape, the presented approach can result in small gaps and overlaps at the turning points, as well as the implement may cross the field boundary. An example is given in figure 12, where the overlap is indicated with black and crossing the boundary is indicated with red. This is due to the tool shape and that the line segments at the critical vertices are not necessarily normal to each other or to the headland boundaries: in fact, if the tool centre point is guided along the swath and lowered and lifted at the turning points, at any critical vertex of the polygon that is not exactly 90 degrees, a small area is left unprocessed and a small area is processed twice. As the headland algorithm is not in the focus of this thesis, this solution was nevertheless considered admissible. The gaps are later corrected with additional parameters that can be tuned during tests in the real-life test environment.

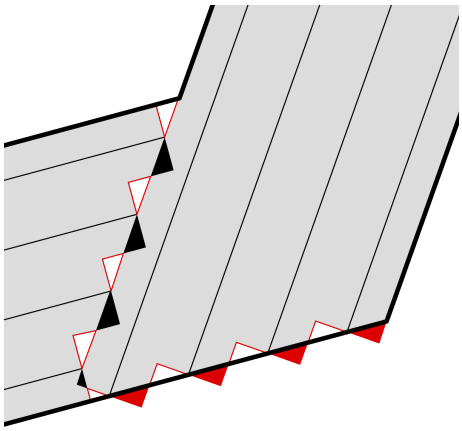


Figure 12: Minor gaps and overlaps may occur in the headland.

Generating the Mainland Swaths

When the mainland is a convex polygonal region the mainland swaths are placed parallel to the longest edge of the mainland boundary. A more optimal method would be to minimize the number of straight parallel paths (and at the same time minimizing the amount of turns) by finding the minimum width of the polygon and placing the swaths perpendicular to the minimum width direction. However this was not an integral part of the goal of this thesis and therefore a simpler method was chosen.

The implement width of the vehicle, which is a parameter in the calculations, determines the spacing of the swaths. The parameter value for the implement width can be changed to yield a small overlap between adjacent swaths. First, the equation of a line is obtained from the two vertices that belong to the longest edge of the mainland boundary. To calculate the endpoints of the first swath centreline, the line is moved towards inside the mainland by one half of the implement width. In a general case when the region is convex, the line intersects the region boundary twice, yielding two endpoints for the swath centreline. For the remaining swaths, the line is moved again towards the same direction as for the first swath but by one tool width at a time, until the line does not intersect the boundary anymore. In case the last swath does not cover the remaining part of the region, one more swath is added to the mainland.

The mainland swaths can be driven in either direction unlike the headland swaths. Therefore when optimizing the next swath every mainland swath is considered twice, once for both directions. The order in which the swaths are eventually driven is computed later during the the cooperative algorithm, where the order is optimized based on an efficiency function. These swath centrelines provide a geometric solution to how to cover the mainland area. These centrelines are later modified during the execution of the cooperative algorithm to consider for example the placement of the GPS receiver on the vehicle. The vertices and the line segments are converted into waypoints where also other parameters such as speed are assigned.

It should be noted that if the swath centreline does not meet the mainland

boundary in a 90 degree angle, as shown in figure 13, the presented approach will result in small gaps and overlaps at the beginning and the end of a swath. However, a parameter can be used to extend the mainland swaths to overlap with the headland to avoid any gaps.

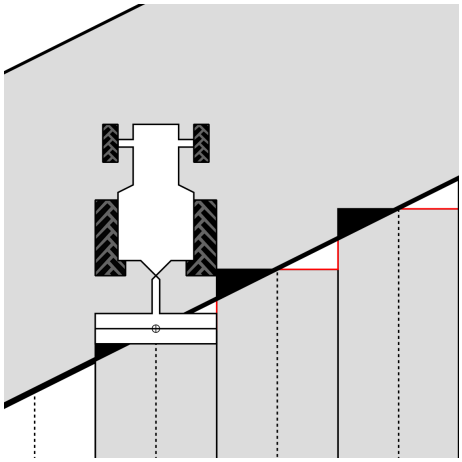


Figure 13: Minor gaps and overlaps may occur in the mainland.

Turn Path Calculation

The vehicles used in this study are non-holonomic and the accessible space outside the field is limited. Therefore, a turn path algorithm is needed to plan the turns between successive swaths. The algorithm that computes the headland turns was implemented as a part of another thesis. The thesis was conducted as a part of the same project as this thesis.

Two functions were provided for the purposes of solving the coverage path for the vehicles: (1) a function `calculatePath` that returns a feasible path between two positions and orientation and (2) a function `calculateTime` that returns a time estimate of the duration of that path. Both functions are called individually.

The function `calculatePath` is used to compute a turn between two swaths. A turn path between two positions can be obtained with a function call `[path, ready] = calculatePath(x1, y1, theta1, v1, x2, y2, theta2, v2, M, id, reset)`, where the input parameters `x1, y1, theta1, v1, x2, y2, theta2` and `v2` refer to the start and end positions of the requested turn including heading and speed, `M` contains the vertex coordinates for the turn area boundary, `id` indicates the vehicle and `reset` is used to reset the calculation. Two outputs are assigned during each function call. Output `ready` indicates the calculation status and output `path` contains the calculation result. When the calculation is ready, the return value `path` is either a list of waypoints if a feasible path was found, or `NaN` if a feasible path was not found.

Since the turn area boundary is given as an input parameter to the function on each function call, it is possible to change or modify the area boundary between function calls. This option is used in one of the algorithms to avoid collisions in advance.

The `calculatePath` function is sliced so that one function call will return in less than 100 ms in normal operation. If the calculation takes more than 100 ms, the function will return even if the computation is unfinished. Once the function is called again it will continue from where it left if the value of input parameter `reset` is `false`. If the value of `reset` is `true`, the calculation will be reset.

The function `calculateTime` is used to obtain a time estimate of the duration of a turn. A time estimate can be obtained with a function call `[time] = calculateTime(x1, y1, theta1, x2, y2, theta2)`, where the input parameters `x1, y1, theta1, x2, y2` and `theta2` refer to the start and end positions of the turn same as for the `calculatePath` function. The time estimate is used in calculating the efficiency of a combination of a turn and a swath.

Solving the Swath Order

The swath centrelines are solved in advance for both the headland and the mainland. The swaths are solved individually for both vehicles because their implements have different dimensions. The first vehicle is allowed to drive any of its swaths from the beginning of the task. For the second vehicle, a swath is available only if it covers an area that has already been processed by the first vehicle. An individual list of the remaining available swaths is maintained for both vehicles. When a vehicle has completed a swath the swath is removed from its list of available swaths.

Unlike the swaths, the order of the swaths, however, is not predetermined. The order of the swaths is solved one by one for both vehicles during the task. A comparison between all available options for the next swath is made at each decision point. The next swath should be chosen for both vehicles before the vehicles reach the end of the previous swath. For both vehicles the list of all remaining available swaths is ordered according to time efficiency. At each decision point, the ordered list is used to choose the best possible option for the next swath.

Only the next swath is considered when the swath order is optimized. Therefore, the optimality of the overall solution is not guaranteed. To choose the next best swath we first determine the efficiency of choosing each of the available swaths. The duration of the swath is considered to be productive time and the duration of the turn is considered to be unproductive time. To maximize the portion of productive time we define the efficiency of a swath as

$$Efficiency = \frac{t_{swath}}{t_{swath} + t_{turn}} = \frac{t_{swath}}{t_{action}} \tag{1}$$

where t_{swath} is the estimated time to complete the swath and t_{turn} is the estimated time to complete the turn. Their sum equals the total estimated time to complete the action t_{action}. To compute this ratio we need the duration of the turn and the duration of the swath. A straightforward estimate for the duration of the swath is obtained from the distance and velocity of each swath waypoint and by assuming an infinite acceleration.

If it is desired to prioritize the headland swaths, an additional parameter can be used to increase the efficiency of the headland swaths by a constant value. Then the

efficiency of an action is defined as

$$Efficiency = \frac{t_{swath}}{t_{action}} + c \tag{2}$$

where c is either a constant value for a headland swath or zero for a mainland swath. Using this parameter can result in an efficiency greater than 1. For the first vehicle the value of c was set to 0.1 to prioritize the headland swaths.

After computing the efficiency of all available swaths they are ordered accordingly. It should be noted that true optimality of the overall route for each vehicle cannot be guaranteed for two reasons. First, only the next swath is considered in the optimization. To ensure a more optimal overall solution the optimization of the next swath should be extended further in the future to include several decision points as in [27]. Second, even if the available swaths were ordered optimally, it cannot be guaranteed that the most optimal swath can eventually be chosen for both vehicles.

It should be noted that the mainland swaths can be driven in either direction. The efficiency is obviously different for each direction since the duration of the turn is different to each end of the swath. Therefore, each mainland swath is considered twice in the list of available swaths and both instances are also removed once the swath is completed.

Start Position and First Swath

Both vehicles can begin their tasks from an arbitrary start position inside the field boundaries. Choosing the first swath is optimized in the same way as choosing the next swath at each decision point. This feature is helpful for testing purposes but for the purpose of comparing different algorithms it should be noted that the start position does affect the overall order of the swaths. This will affect the overall optimality of the solution but also the possibility that at some point both vehicles cannot be routed simultaneously.

Coverage Map Update

A map of the unprocessed area is maintained for both vehicles. The area is represented by a polygon consisting of several vertices, one polygon for each vehicle. The status updates of both vehicles are received at approximately even time intervals. The area that was processed between two consecutive status updates is calculated based on the implement working position, GPS data and the dimensions of the vehicle and the implement. At each status update this calculation is done for both vehicles. The unprocessed area for the second vehicle is used only to monitor the task status. The unprocessed area for the first vehicle is used also to update the available swaths for the second vehicle.

To track the unprocessed area we begin with a polygon that equals the boundary of the field. The polygon is modified at each status update by subtracting the area that was processed between the newest and the previous status update. The implement position between the two status updates is determined from the hitch

level values. A polygon is calculated for the processed area only if the implement was lowered. This polygon is always a trapezoid approximation of the area that was actually processed between the two status updates. The trade-off between a long and a short time interval between two consecutive status updates is between mitigating the impact of GPS noise on the resulting unprocessed area boundary and increasing the accuracy of the calculation.

The calculation of the processed area between two status updates is pictured in figure 14. The GPS coordinates and heading define the position (x, y) and orientation θ of the vehicle. The endpoints of the implement are calculated based on the placement of the GPS receiver on the vehicle and the dimensions of the vehicle and the implement. The distance between the two positions in two consecutive status updates depends on the time interval t and the velocity and acceleration of the vehicle during that time. The updated unprocessed area is obtained by subtracting the resulting trapezoid, shown in red, from the unprocessed area polygon.

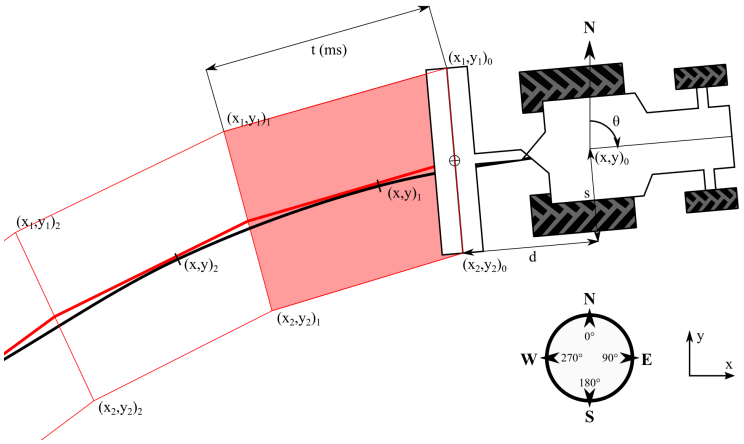


Figure 14: Approximation of the area processed between two status updates.

Available Swaths for the Second Vehicle

The second vehicle is allowed to drive only those swaths that cover an area that the first vehicle has already processed. Continuous update of the progress of the first task is needed to update the available swaths for the second vehicle. Instead of tracking the area that has been processed we monitor the area that is unprocessed. Then the available swaths for the second vehicle are the ones that do not overlap the unprocessed area of the first vehicle. The polygonal coverage area for each swath is determined from the previously calculated swath centrelines. If the coverage area of a swath does not overlap the unprocessed area of the first vehicle, the swath is marked as an available option for the next optimization.

When the swaths for both vehicles are placed in the same orientation it is most likely that new swaths for the second vehicle will be available only when the first vehicle has completed a swath, not during the first vehicle is still processing the swath. Therefore the available swaths for the second vehicle are revised only when the vehicles have reached the end of the previous swath.

Conflict Detection

A means to detect possible collisions is necessary when the path for each vehicle is planned individually. For safety reasons also a final validation is performed on every solution before forwarding new waypoints to the vehicles. A simple method of conflict detection was implemented with polyline buffering and a boolean operation on polygon overlap. Both algorithms are readily available in Matlab.

To determine whether two paths are in conflict the path of each vehicle is buffered with a collision zone based on the vehicle dimensions and a safety margin parameter. The vehicle dimensions and the safety margin parameter define the buffer distance. Given the path and the buffer distance, the buffer polygon is computed by moving a circle of radius equal to the buffer distance along the line segments defined by the path. If the resulting polygons overlap the paths are considered to collide.

An example of a conflict of two paths is shown in figure 15. The original paths are shown with red and blue polylines. The corresponding collision zone polygons are shown in red and blue. The overlap is determined by a polygon overlap function readily available in Matlab.

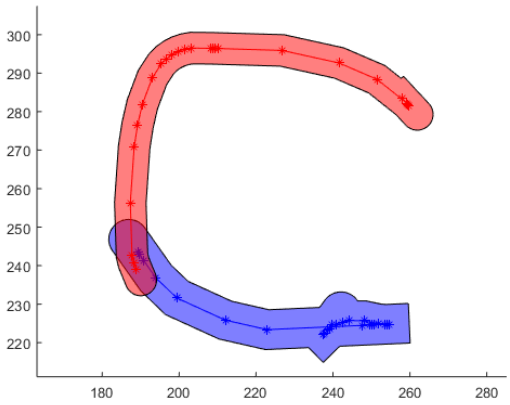


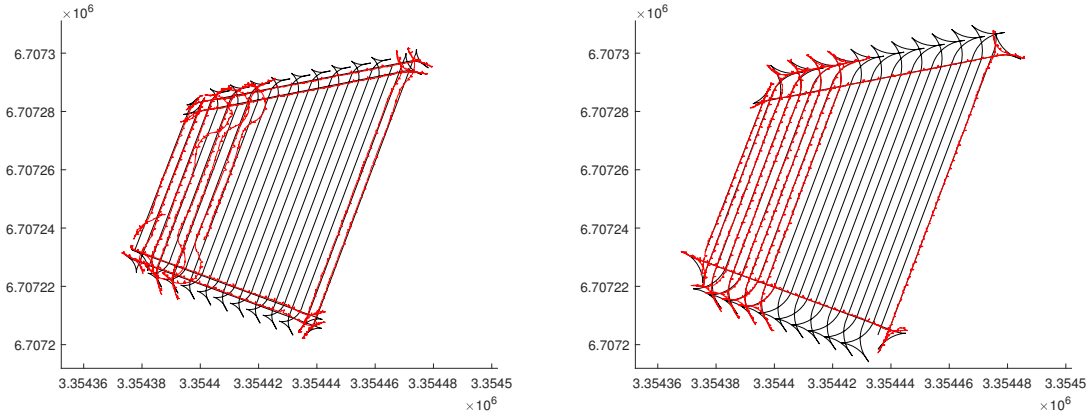
Figure 15: An example of collision detection on two conflicting paths.

Consideration of Real-life Constraints

The coverage path can be first solved for a point mass with a fixed tool, but these assumptions do not hold in real life. Several measurements are needed to calculate the coverage path correctly. In real life there are various error sources and nonidealities: the tool cannot be lowered or lifted instantly, all the dimensions or the minimum turning radius of the vehicle can not be measured exactly, a good driving speed may depend on the field conditions and so on. Therefore, several tuning parameters and additional processing routines were included in the algorithms.

For example, the minimum turning radius of a vehicle is a tunable parameter in the algorithms. The value for the turning radius needs to be close enough or larger than the real minimum turning radius of the vehicle to avoid problems with navigation. The turn path algorithm was not implemented as a part of this thesis,

however, it has an impact on the coverage result as is illustrated in figure 16. The planned path is shown in black whereas the actual GNSS data is shown in red. It is visible from figure 16a that the test vehicle has major difficulties of recovering from a turn when a route is planned with a turning radius of 7 meters compared to a turning radius of 12 meters in figure 16b. No other relevant parameters were altered between the two test drives.



(a) Turning radius of 7 meters.

(b) Turning radius of 12 meters.

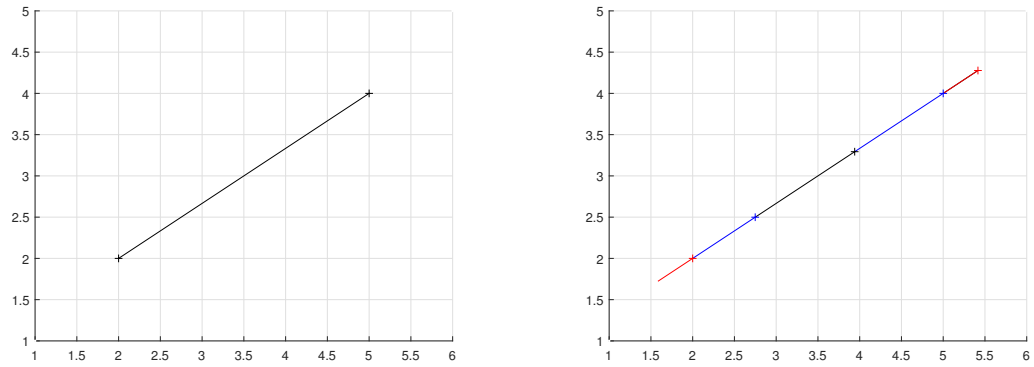
Figure 16: Impact of turns and the turning radius parameter on the coverage result (planned path in black and actual GPS data in red).

To ensure that the tool is lowered correctly at the beginning of a swath and lifted correctly at the end of a swath, two additional entry waypoints and two additional exit waypoints are generated for each swath. Each swath then begins with two entry waypoints, folowed by the waypoints of the main part of the swath, and ends with two exit waypoints. The main part of the swath can consist of several waypoints, especially if the swath is curved.

The speed setpoint for all entry and exit waypoints is lower than the speed setpoint for the waypoints of the main part of the swath. The tool is lowered during the second entry waypoint and lifted during the second exit waypoint. The tool is therefore safely lifted during lower driving speed and only after and before a turn between successive swaths.

The length of the entry and exit waypoints can be adjusted as well. The length can be adjusted for example to accomodate different speeds and acceleration limits of the vehicles. This is also useful for tools that process the soil in two phases. For example, the disk cultivator used in the tests of this study has two rows of disks to process the soil. Therefore, the tool may need to be lowered earlier and lifted later at the beginning and the end of a swath.

An example of entry and exit waypoints for a simple straight swath is shown in figure 17. The swath is very short for demonstration purposes. In the example the first entry waypoint and the last exit waypoint are generated outside the original swath.



(a) Original swath.

(b) Added entry and exit waypoints.

Figure 17: Additional entry and exit waypoints in the beginning and the end of a straigth swath.

5.3 Mission Planner

Two different algorithms were developed to solve the cooperative coverage path planning problem. A Matlab class called *Mission Planner* was developed in Matlab R2018a and the algorithms were implemented as class methods of the Mission Planner class. The Mission Planner class implementation can be used from Matlab or Simulink but it can also be assembled into a Dynamic Link Library (DLL) to be used from the Mission Operator C# program. A set of interface functions and a function call proxy were written to be assembled into the DLL. To be able to use the Mission Planner class via the DLL, the call proxy holds a persistent instance of the Mission Planner class. The purpose of all the interface functions is to forward the function call they receive to the call proxy, which in turn calls the corresponding class methods of the persistent instance of the Mission Planner class with the given input values.

The utility functions that were previously described in chapter 5.2 are used in the Mission Planner class as a part of the implementation of the cooperative coverage path planning algorithms.

Initialization

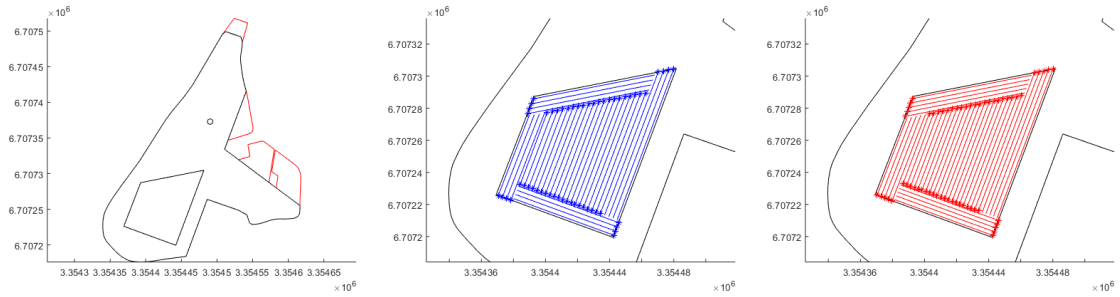
The Mission Planner class is initialized with two different maps that describe the geometry of the field, and several parameters that describe the vehicles and allow modifying the algorithm behaviour. Two different maps are used so that different regions can be defined for the work area and the turn area on the field. However the work area must be either equal to the turn area or completely contained inside the turn area.

The real test field that was used in this study is located in Vihti, Finland. For testing the algorithms, several smaller test fields were defined inside the boundaries of the real test field. Some of the boundaries of some of the test fields are artificial so that in reality the vehicles are able to drive outside the boundaries. Each test field is a polygon that is defined by its vertices. All the field polygons are stored in separate

shapefiles in the same directory. In the shapefiles the polygon vertices are stored in global YKJ coordinates. For the computations the coordinates are transformed into local coordinate system with values in the range of $[0, 1000]$.

A script is used to load the shapefiles from the directory and to compile them into one Matlab data file. The Matlab data file is loaded from the file system when an instance of the Mission Planner class is created. The content of the file is saved as a class property of the Mission Planner class instance. An instance of the Mission Planner class has access to all the available field shapes loaded from the data file. All of the test fields are associated with a unique identifier that is used to choose the field area and turn area for the task.

A visualization of the computation during class initialization is shown in figure 18. The real test field boundaries are shown in figure 18a with a trapezoidal test area defined inside the field boundaries. It is possible to define the turn area separately so that the vehicles can access the area outside the test are boundaries during turns. After the test area boundaries have been defined, the swath centrelines for both vehicles are precomputed during class initialization. For the trapezoidal test area shown in figure 18a, an example of the swath centrelines for both vehicles are shown in figures 18b and 18c.



(a) A trapezoidal test area. (b) Swaths for 1st vehicle. (c) Swaths for 2nd vehicle.

Figure 18: A test area inside the real field boundaries and the precomputed swath centerlines for both vehicles.

In addition to the field geometry, several parameters are needed to initialize the task. Most of the parameters are used to define the vehicle characteristics and to modify their individual behaviour during the task. Several parameters are used in the algorithm computations. The parameter values need to be assigned before the beginning of the program execution.

Time-slicing the Algorithms

Both algorithms that were developed in this thesis are based on the idea of computing short simultaneous paths that do not overlap, and then scheduling them in real-time. The real-time computation aspects require that the method calls to the algorithms can be timed with a constant interval timer for example every 200 ms. To meet the real-time requirements the algorithms are time-sliced so that one method call will return in less than 100 ms in normal operation even if the computation is unfinished.

Also the utility functions need to consider this time constraint and therefore all the utility functions that could require over 100 ms for the computation are time-sliced as well.

All functions whose execution could take over 100 ms are time-sliced simply by timing the execution. The functions use an internal timer and when the timer value exceeds a predetermined threshold, all intermediate results are saved and the function returns. The function will continue from where it left once it is called again. All time-sliced functions implement a similar interface where one input value is used to reset the calculation.

Cooperation

MissionPlanner is a Matlab class that holds access to all the necessary data including parameter values and task status information. The cooperative coverage path planning algorithms are implemented as methods of the MissionPlanner class. Two alternate algorithms were developed to enable cooperation. Both algorithms are based on the idea that even without planning explicit trajectories (with regards to time), the vehicles are able to work simultaneously if for both vehicles only one turn and swath are scheduled at a time. Then it is necessary to only ensure that those paths, including sufficient safety margins, do not overlap, and that before scheduling the next path, both vehicles have completed their previous paths. The full solution to complete a field is therefore not calculated at once but one turn and swath for each vehicle at a time. The scheduling of each action, one turn and one swath for both vehicles, is done by the Mission Operator in real-time.

Cooperative Algorithm A

Two alternate algorithms were developed to enable cooperation. The implementation of Algorithm A (and Algorithm B) is based on a state machine. Calling the class method `Algorithm_A(breakOptimization, restartOptimization)` equals advancing the Algorithm A by one step. The method call always returns three values `ready`, `waypoint_vehicle1` and `waypoint_vehicle2`.

The algorithm is time-sliced so that one method call will return in less than 100 ms in normal operation. During one method call the algorithm will advance not more than to the next state of the state machine. If the calculation during a state takes more than 100 ms, the method will return nevertheless, and continue from where it left once the method is called again. The method always returns between state transitions. Therefore, one function call to the algorithm equals one state of the state machine at most.

There are 9 states in the Algorithm A state machine. One cycle of the algorithm, from the beginning of State 1 to the end of State 9, yields the next action for both vehicles. If both vehicles can be routed, the actions for both vehicles consist of waypoints for a turn and a swath. If there was a conflict that could not be solved, the other vehicle will not receive any new waypoints. During the execution of the next action that vehicle will wait where it is currently located. Which vehicle has to

wait depends on the reroute policy. The idea behind Algorithm A can be simplified as follows

(1) For both vehicles, optimize the next swath based on efficiency

(2) For both vehicles, choose the best swath

(3) For both vehicles, calculate turn from the current location to the beginning of the swath

(4) Validate that the paths do not collide

- in case of collision, reroute (choose the next best swath for) either vehicle according to the reroute policy (reiterate until solution is found or no possible solutions are left)
- in case of no collision, route both vehicles as per current solution

(5) In an unsolved conflict, either vehicle waits according to the chosen reroute policy

In Algorithm A, a candidate action for both vehicles is calculated independent of each other. This approach could easily result in conflicts that need to be solved by re-routing either of the vehicles. Obviously there are various options of how to perform re-routing, including which vehicle will be re-routed and which vehicle is prioritized in case of an unsolved conflict. In case of a conflict a so called *re-route policy* is applied.

The state diagram of Algorithm A is presented in figure 19. The algorithm is activated with the first call to method `Algorithm_A(...)` and it begins in State 1. In State 1, the next possible swaths for both vehicles are ordered according to optimality as defined in equation 2. The optimization phase can be interrupted if the method is called with value `true` for parameter `breakOptimization` while the algorithm is in State 1. Interrupting the optimization may result in a suboptimal order of the next possible swaths. Otherwise the optimization is continued until the optimal solution for both vehicles has been found.

The waypoints for the candidate solutions for both vehicles are calculated in State 2. This computation can easily take more than 100 ms and the state machine often remains in State 2 for multiple method calls. The path for the first vehicle is calculated first, and after that the path for the second vehicle. When both paths are ready the state machine transitions to State 3. Since the paths were calculated independently in State 2 they can obviously collide. In State 3 the algorithm validates that no collisions are encountered if these paths are assigned to the vehicles simultaneously. Provided that the collision zones of the paths do not overlap the algorithm proceeds to State 7.

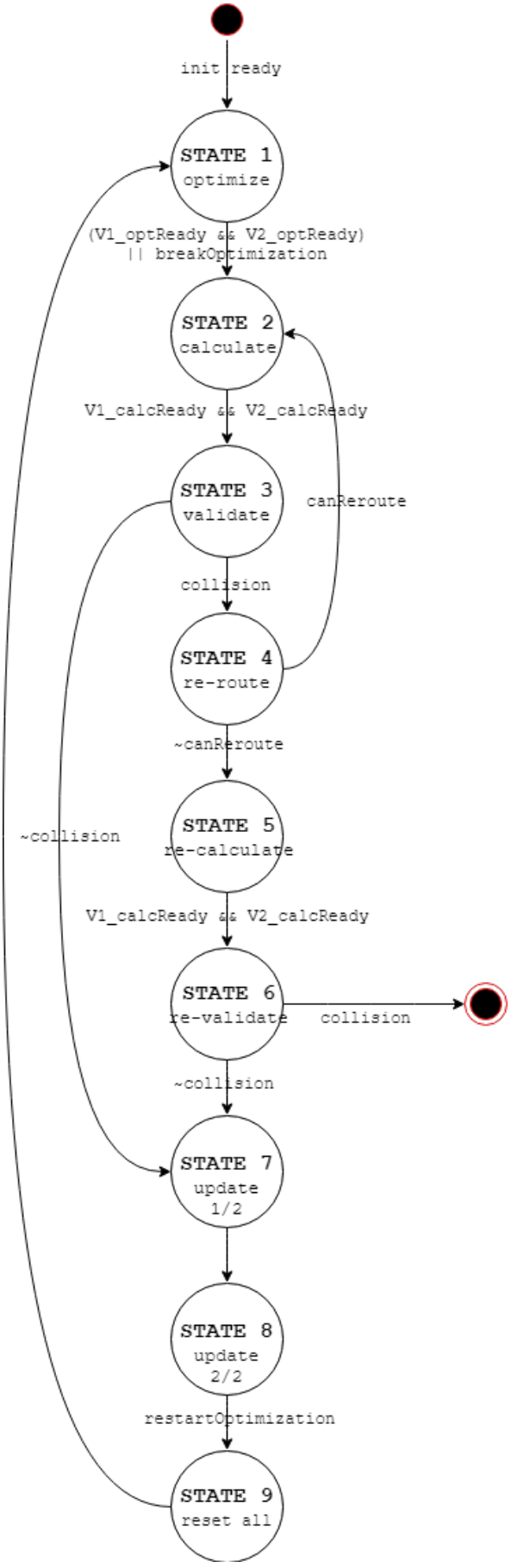


Figure 19: State diagram of Algorithm A.

If, however, a conflict is detected in State 3 then the algorithm transitions to State 4 where either of the vehicles is rerouted. The rerouting is performed according to a *rerouting policy* that defines which vehicle is rerouted and which vehicle is eventually prioritized if two compatible paths cannot be found. The reroute policies are identified with two-digit identifiers where the first digit denotes the vehicle that is rerouted and the second digit denotes the vehicle that is prioritized. Digit 1 is for the first vehicle, digit 2 is for the second vehicle and digit 3 means 'to alternate'. For example the identifier 31 as in Algorithm A31 indicates a rerouting policy where both vehicles are rerouted in turns and the first vehicle is prioritized if no feasible combination of two paths can be found. From State 4 the algorithm returns to State 2 or proceeds to State 5 if rerouting is unsuccessful.

In State 5 the vehicle which is prioritized is rerouted back to the most efficient swath and the action of the other vehicle is cancelled. This means that the other vehicle needs to wait during the next turn and swath. A final validation is performed in State 6. If the path for the prioritized vehicle overlaps the position of the vehicle whose action was cancelled, the algorithm is in a deadlock.

After the solution has been validated in State 3 or in State 6, the task status is updated accordingly. The available swaths for the 2nd vehicle are updated in State 7. The update phase is divided into two states because updating the available swaths for the 2nd vehicle may take more than 100 ms and the calculation needs to be sliced. In State 8 the waypoints for both vehicles are updated and marked as ready to be sent forward. The return value `ready` is assigned `true` in State 8.

The algorithm remains in State 8 until the next function call is made with value `true` for parameter `restartOptimization`. In State 9 all variables that concern one cycle of the state machine are reset. The transition back to State 1 requires no specific condition.

Cooperative Algorithm B

Two alternate algorithms were developed to enable cooperation. The implementation of Algorithm B (and Algorithm A) is based on a state machine. Calling the class method `Algorithm_B(breakOptimization, restartOptimization)` equals advancing the Algorithm B by one step. The method call always returns three values `ready`, `waypoint_vehicle1` and `waypoint_vehicle2`.

The algorithm is sliced so that one method call will return in less than 100 ms in normal operation. During one method call the algorithm will advance not more than to the next state of the state machine. If the computation during a state takes more than 100 ms, the method will return nevertheless, and continue from where it left once the method is called again. The method always returns between state transitions. Therefore, one function call to the algorithm equals one state of the state machine at most.

There are 10 states in the Algorithm B state machine. The states are numbered so that State 1, State 7, State 8 and State 9 have the same number as in Algorithm A. These states are identical in both algorithms as in the events in these states are the same. Because of this numbering, State 10 in Algorithm B is not after State 9

but between State 2 and State 3. One cycle of the algorithm, from the beginning of State 1 to the end of State 9, yields the next action for both vehicles. If both vehicles can be routed, the actions for both vehicles consist of waypoints for a turn and a swath. If there was a conflict that could not be solved, the second vehicle will not receive any new waypoints. During the execution of the next action the second vehicle will wait where it is currently located. The idea behind Algorithm B can be simplified as follows

(1) For both vehicles, optimize the next swath based on efficiency

(2) For both vehicles, choose the best swath

(3) For the 1st vehicle, calculate the turn from the current location to the beginning of the next swath

(4) For the 2nd vehicle, modify the turn area to exclude the collision zone of the path for the first vehicle

(5) Validate that the swath for the 2nd vehicle does not collide with the path of the 1st vehicle

- in case of collision, choose the next best swath for the 2nd vehicle (reiterate until a solution is found or no possible solutions are left)
- in case of no collision, route both vehicles as per current solution

(6) For the 2nd vehicle, calculate the turn from the current location to the beginning of the next swath

(7) Validate that the paths do not collide

- in case of collision, cancel routing for the 2nd vehicle
- in case of no collision, route both vehicles as per current solution

In Algorithm B, the path for the 1st vehicle is calculated independent of the path of the 2nd vehicle. The path for the first vehicle, however, is considered when computing the path for the second vehicle. To avoid collisions, the path for the first vehicle is excluded from the available turn area for the second vehicle. This approach is less likely to result in conflicts than the approach in Algorithm A.

The state diagram of Algorithm B is presented in figure 20. In the diagram, the first vehicle is referred to as *V1* and the second vehicle is referret to as *V2*. The algorithm is activated with the first call to method `Algorithm_B(...)` and it begins in State 1. In State 1, the next possible swaths for both vehicles are ordered according to optimality as defined in equation 2. The optimization phase can be interrupted if the method is called with value `true` for parameter `breakOptimization` while the algorithm is in State 1. Interrupting the optimization may result in a suboptimal order of the next possible swaths. Otherwise the optimization is continued until the optimal solution for both vehicles has been found.

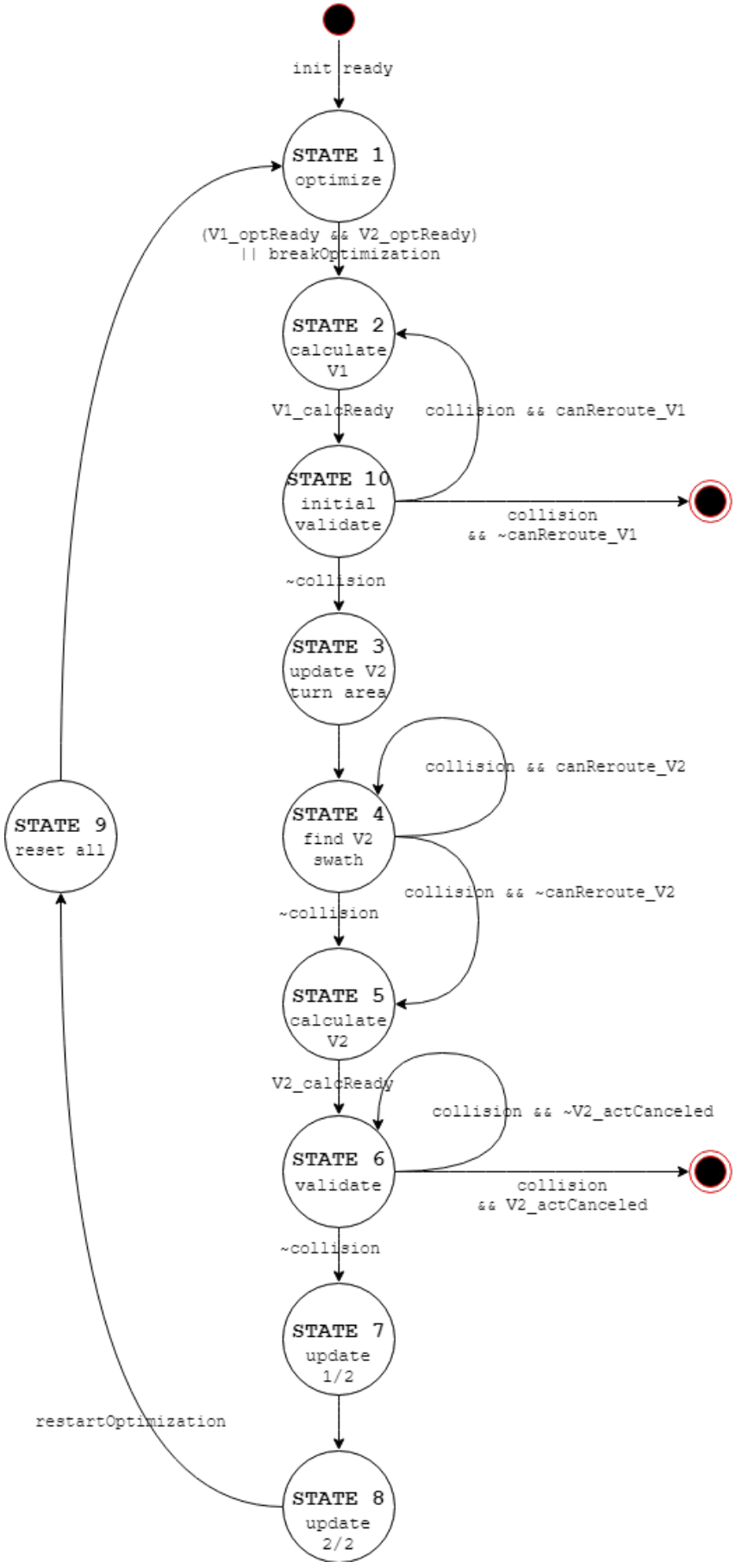


Figure 20: State diagram of Algorithm B.

The waypoints for the path for the first vehicle are calculated in State 2. This computation can easily take more than 100 ms and the state machine often remains in State 2 for multiple method calls. When the path is ready the state machine proceeds to State 10. The initial validation in State 10 is to ensure that the path for the first vehicle does not overlap with the current location of the second vehicle. If the path is unobstructed the algorithms proceeds to State 3.

In State 3 the collision zone of the first vehicle's path is removed from the available turn area for the second vehicle. Consequently the turn path calculation for the second vehicle will evade this area if necessary. In State 4 the algorithm finds a feasible swath for the second vehicle. If no feasible swath exists, the action for the second vehicle will be cancelled. This means that the second vehicle needs to wait during the next turn and swath.

The waypoints for the path for the second vehicle are calculated in State 5. This computation can easily take more than 100 ms and the state machine often remains in State 5 for multiple method calls. When the path is ready the state machine proceeds to State 6. The final validation in State 6 is to ensure that the paths are not in conflict. If a collision is detected the action for the second vehicle can still be cancelled, however, it unlikely to change anything due to the initial validation in State 10.

If the solution was been validated as feasible in State 6, the task status is updated accordingly. First the available swaths for the 2nd vehicle are updated in State 7. The update phase is divided into two states because updating the available swaths for the 2nd vehicle may take more than 100 ms and the calculation needs to be sliced. In State 8 the waypoints for both vehicles are updated and marked as ready to be sent forward. The return value `ready` is assigned `true` in State 8.

The algorithm remains in State 8 until the next function call is made with value `true` for parameter `restartOptimization`. In State 9 all variables that concern one cycle of the state machine are reset. The transition back to State 1 requires no specific condition.

5.4 Mission Operator

The main program for the real-life test set-up called *Mission Operator* was developed in C# with .NET Framework. The Mission Operator is responsible for supervising, scheduling and controlling the overall mission including all communications with the vehicles and executing the Mission Planner cooperative coverage planning algorithm.

A .NET assembly of the Mission Planner was compiled with Matlab Library Compiler. The resulting assembly is a dynamic-link library (DLL), which is a shared library concept in the Microsoft Windows. A data conversion API is needed to marshal and format data across the .NET and Matlab code boundary. The MWArray API (Application programming interface) provides marshalling and formatting for all basic Matlab types. The MWArray data conversion classes allow most native .NET value types to be passed directly as parameters to the functions in the Matlab DLL. The MWArray API is included as a DLL within MATLAB Runtime. To use the Mission Planner in the C# program, a reference is added to both the Mission

Planner DLL and the MWArray API DLL in the Mission Operator C# solution.

The target system does not require a licensed copy of Matlab to use the DLL built from Matlab code with Matlab Library Compiler, however, the MWArray API requires the MATLAB Runtime to be installed on the target machine.

A simplification of the overall operation of the Mission Operator program is presented in figure 21. This diagram describes the real-time operation of the Mission Operator program during the real-life tests of the cooperative path planning algorithms.

Scheduling

With the approach that was adapted to solve the cooperative coverage path planning task it is essential to carefully consider the timing constraints. The Mission Planner solves simultaneous non-overlapping paths for the vehicles but does not schedule them. In addition to implementing the communication interface to the vehicles over UDP, another main task for the Mission Operator program is to schedule the simultaneous paths for the two vehicles in real-time.

The Mission Planner class method that corresponds to the cooperative path planning algorithm is called in a 200 ms cycle in the Mission Operator. The class methods for the path planning algorithms have two input values, `breakOptimization` and `restartOptimization`, to control the progress of the algorithm. Each method call always returns three outputs `ready`, `waypointVehicle1` and `waypointVehicle2`. The value for output `ready` is `true` only if the algorithm has reached the end of the algorithm cycle and the next action for the vehicles is ready. Then the algorithm remains in the last state of the state machine until it receives a restart command from the Mission Operator.

The output values `waypointVehicle1` and `waypointVehicle2` always contain one waypoint for each vehicle. If there are no new waypoints to be forwarded to the Mission Operator the last waypoint is returned again. Then if the index of the last received waypoint is equal to the index of the previously received waypoint, all waypoints that have been calculated in the Mission Planner have already been received in the Mission Operator. If the new waypoint that was received from the Mission Planner is the same as the previous one, then the last batch of waypoints (turn and swath) were received and we don't expect new waypoints from the Mission Planner until we command the algorithm to solve the next turn and swath.

In the Mission Operator, new waypoints are requested from the Mission Planner only when necessary. The algorithm is ready to proceed to compute the next action when the value for output `ready` is `true`. To proceed to the next algorithm cycle the input value `restartOptimization` is set to `true` for one method call and then set to `false` again. The time it takes for the Mission Planner to compute the next action is indeterminate. When the next action has been computed successfully, the new waypoints are returned one by one in the output values `waypointVehicle1` and `waypointVehicle2`.

When the waypoints for the next action have been received in the Mission Operator, they are scheduled for execution by sending them to the vehicles at the

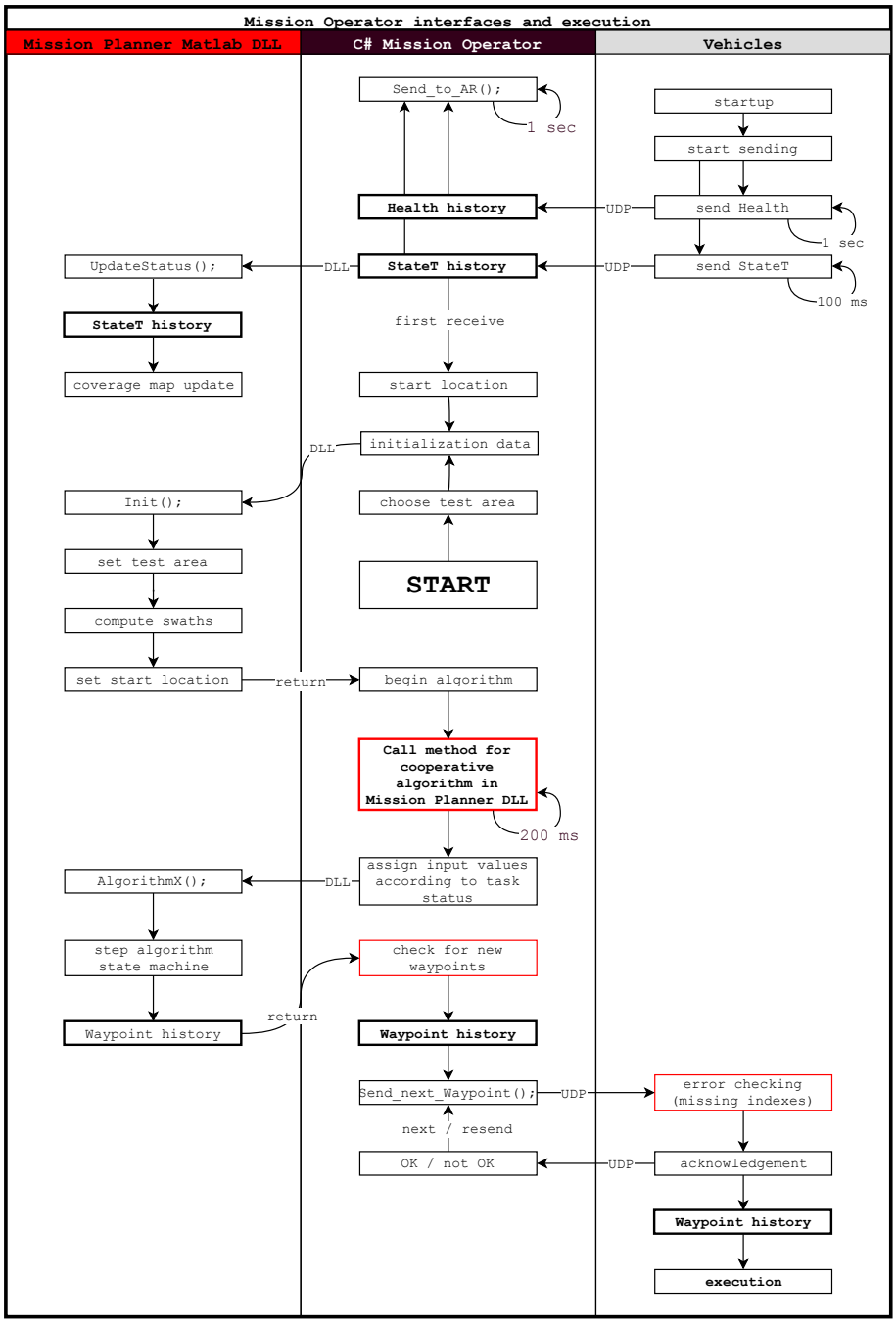


Figure 21: A simplification of the Mission Operator interfaces and execution.

right time. To safely schedule the new waypoints for the vehicles, the Mission Operator verifies that the vehicles have completed all the previously sent waypoints. The waypoints are sent one by one for both vehicles, ensuring that a positive acknowledgement is received for each waypoint before sending the next one.

If the paths are not scheduled in this way, a collision could happen between the previous and the current action because the Mission Planner validates only that the

two simultaneous paths that it computes do not overlap. Non-real-time collision detection is not enough if the vehicles have different speed, if their paths are different length, or an error or an interruption is encountered. Because the Mission Planner computes paths, not trajectories, it is impossible for the Mission Planner alone to fully verify that the vehicles cannot collide.

The input value `breakOptimization` is used only if the vehicles are close to completing the previous action before the optimization of the swath order is complete. However this input value is rarely used since the current optimization procedure is not very demanding computationally.

Also the vehicle status feedback is updated to the Mission Planner with the same 200 ms interval as the Mission Operator calls the Mission Planner path planning algorithm. The status feedback is used to update the coverage map of both tasks. The coverage map of the first vehicle defines the available swaths for the second vehicle.

Guidance

During the mission, individual routes are planned for both vehicles. The routes are represented with an ordered sequence of waypoints, where each waypoint contains a coordinate pair that indicates the target location easting and northing in YKJ. The waypoints form a polyline where the coordinates of two successive waypoints define the endpoints of a line segment. The routes for both vehicles are planned online, and therefore new waypoints are continuously added to the end of the routes. An index number is included in all waypoints to ensure their correct order.

In addition to the coordinate pair and an index number, a waypoint contains values for heading offset, speed, acceleration limit and implement working position. The heading offset, speed, acceleration limit and implement position are set for the duration of the segment. The implement position value indicates whether the tool should be in use during the segment. All data contained in a single waypoint is shown in table 3.

Table 3: Waypoint data from Mission Operator (*Waypoint*).

Data	Definition
Index	Index number of the waypoint
X	Target location easting in YKJ
Y	Target location northing in YKJ
Heading offset	Target offset of vehicle heading to segment direction
Speed	Target speed during the segment
Acceleration limit	Acceleration limit during the segment
Implement position	Implement position during the segment (up, down)

The first waypoint of a route is a special case since there are no previous waypoints. Therefore, for the first waypoint, all the other values than the easting and northing of the coordinate pair are zero. The endpoints of the first segment of the route is

defined by the coordinate pairs of the waypoint zero and the first waypoint. An illustrative example of the beginning of a route is shown in figure 22.

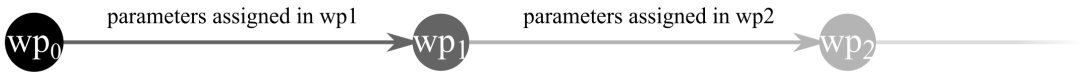


Figure 22: Example of waypoints at the beginning of a route.

For headland turns and other parts of the path with curvature the curve can be approximated with multiple waypoints with small enough intervals. For example a 90 degrees arc could be described with 18 waypoints each covering 5 degrees of the arc. Any such special cases require particular attention, including entering and exiting the headland when changes in speed are sometimes substantial.

The number of waypoints that are needed to cover an entire field depends mainly on the field size and complexity, including the number and type of turns in the headland. A limit of 50000 waypoints was set in the Mission Operator for the total number of waypoints in the route for each vehicle.

The guidance does not set any explicit constraints on the time of arrival with regards to reaching a specific waypoint. The travel along a segment is executed within the speed and acceleration limit that are assigned in the waypoint. The exact time it takes to travel a segment depends on the navigation of the vehicle.

The waypoint messages are sent over UDP. Because UDP gives no guarantee of neither delivery nor order of arrival, an acknowledgement is expected from the vehicles for each waypoint message.

Table 4: Waypoint acknowledgement from the vehicles (*Ack*).

Data	Definition
Index	Index number of the corresponding waypoint that was received
Ack value	Value 0 for ok, values 1 to 4 for error codes

Feedback

Status feedback from the vehicles is needed to monitor the progress of the tasks and to plan the following routes accordingly. The vehicles update their current status to the mission operator approximately every 100 ms.

The feedback data includes, among others, the current location of the vehicle and the implement position. This data is used to calculate the field area that has already been processed. The time left and distance left data are used to estimate when new waypoints need to be sent to the vehicles. All status data from the vehicles is shown in table 5.

Table 5: Status data from the vehicles (*StateT*).

Data	Definition
X	Current location easting in YKJ
Y	Current location northing in YKJ
Latitude	Current location latitude in WGS84
Longitude	Current location longitude in WGS84
Altitude	Current location altitude in WGS84
Heading	Current heading in WGS84
GPS quality	Current GPS fix quality according to NMEA GGA
Speed	Current speed
Steer angle F	Current front steering angle
Steer angle R	Current rear steering angle
Hitch level	Current hitch level (implement position 0-100)
Segment	Current waypoint under execution
Time left	Approximated time left to reach the last waypoint received
Distance left	Distance left to reach the last waypoint received
Autopilot on	Autopilot on (true, false)
GPS valid	GPS signal valid (true, false)

For the AR application, some additional data is received from the driverless tractor. This data is not used in the Mission Operator or in the cooperative path planning algorithms, but only forwarded to the AR application. This data includes auxiliary status information such as battery level, amount of remaining fuel, and motor temperature and RPM.

Communication

Three messages are used for communication between the Mission Operator and the vehicles. An individual waypoint message is sent from the Mission Operator to both vehicles and the vehicles respond with an acknowledgement. A third message, one from each vehicle, is sent from the vehicles to the Mission Operator. This message contains the status information, or feedback, from the vehicles.

During the mission, an individual route is generated online for both vehicles. The routes are represented with an ordered sequence of waypoints. In addition to the coordinates (x,y) and an index number, a waypoint contains values for heading offset, speed, acceleration limit and implement working position. The waypoints form a polyline where the coordinates of two successive waypoints define the endpoints of a line segment. The heading offset, speed, acceleration limit and implement position are set for the duration of the segment. While new waypoints are available, a waypoint message is sent to the vehicles every 100 ms. One message contains one waypoint and separate messages are sent for both vehicles.

An additional message is forwarded through the Mission Operator software to an augmented reality (AR) application. The application was developed as a part of the project in another thesis work. Vehicle data is provided to the AR application in a separate UDP message. The message is combined from the status information

of both vehicles and an additional health message from the driverless vehicle. The application that runs in a set of AR glasses provides a user interface for the human operator.

Both vehicles send their status messages to the Mission Operator every 100 ms. The moment of transmission is not synchronised between the vehicles and therefore the messages may arrive at different times. The status messages from the vehicles are not responded with any acknowledgement, however, the time of the last received status message is monitored to ensure the vehicles are online and operational.

In addition, the health information of the driverless tractor is received as a UDP message. This data is combined with the status messages of both vehicles and forwarded to the AR application via UDP.

The communication between the vehicles and the Mission Operator was implemented over Ethernet. Two options were considered for communication protocol. User Datagram Protocol (UDP) is a connectionless communication protocol to send messages over an IP network with a minimum of protocol mechanism. UDP provides an unreliable, transaction oriented data transfer service such that neither delivery, order of arrival nor duplicate protection of the datagrams are guaranteed [91].

An alternative for applications requiring ordered reliable delivery of data over an IP network is the Transmission Control Protocol (TCP). Unlike UDP, TCP is a connection-oriented protocol that uses flow control, sequence numbers, acknowledgements and timers to ensure reliable, correct and ordered data transfer between applications [92]. UDP, having minimal protocol services, typically has shorter latency and gives higher throughput than TCP.

Mainly due to shorter latency and simple implementation, UDP was chosen for data transmission between the vehicles and the main computer running the Mission Planner. All data from the Mission Planner to the vehicles and vice versa is sent via UDP messages.

Since UDP protocol does not provide any confirmation of delivery, a simple acknowledgement protocol was implemented for the waypoint messages. While new waypoints are available, a UDP message is sent to the vehicle every 200 ms. The vehicles respond with an acknowledgement indicating whether the waypoint was received correctly: the vehicles expect to receive the waypoints in an ascending order according to the waypoint index. If the response indicates that the index was not empty or an index was skipped, another waypoint is sent accordingly.

Graphical User Interface

A simple graphical user interface (GUI) was developed for the Mission Operator. The main purpose of the GUI is to visualize the status of the operation. An option to write a log file of the vehicle status histories was implemented as a part of the Mission Operator program. A command to write the current status history of either vehicle to a log file can be given from the GUI. The GUI is shown in figure 23.

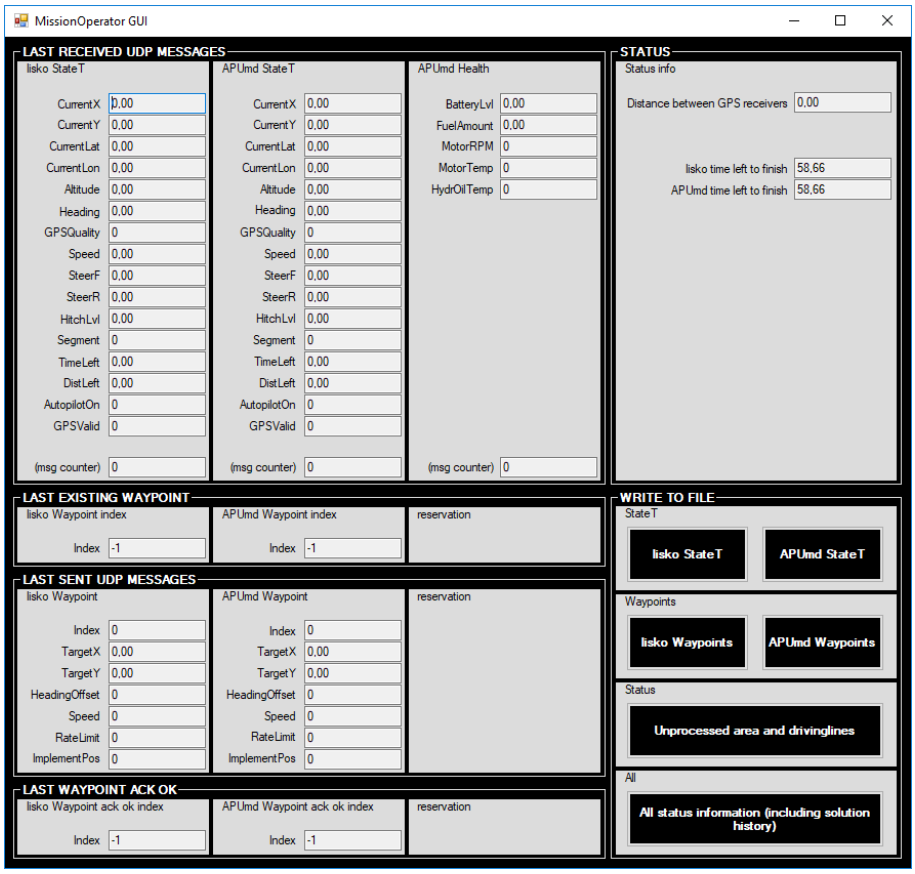


Figure 23: Mission Operator graphical user interface.

6 Simulation Environment

Implementation of a simulation environment was originally a part of the goals of this thesis, however, it could not be completed in time due to technical issues and time constraints. Two different simulations were planned to serve as a testing environment during the algorithm development. The necessity for two separate simulations stems from two test purposes. First, the algorithm testing could be performed on a non-real-time basis for fast initial evaluation. Second, it is sensible to test the correct operation of all interfaces of the real-time guidance before operating with real vehicles at the test premises. In addition, testing the algorithms in a simulation is possible even when conducting real-life tests outside is challenging or infeasible due to seasonal constraints. For one, a non-real-time simulation was designed for Mission Planner algorithm testing and initial validation. The other, a type of a Hardware in the Loop (HIL) simulation, was designed to test the system architecture and interfaces.

Mathworks Matlab and Simulink environment was used to implement both simulations. The implementation of the navigation and the kinematics of the tractors are the same in both simulations. Almost identical Simulink models were used to generate the navigation code for the actual vehicles. Their Simulink implementations were provided by the thesis instructor, and only minor modifications were made for simulation purposes as a part of this thesis.

The main difference between the two simulations is the interface to the Mission Planner algorithm. The implementation of the simulations is described briefly in chapter 6.1 for the non-real-time simulation and chapter 6.2 for the HIL simulation. In addition, a visualization for the simulations was implemented in Matlab code for visual validation. The visualization implementation is described shortly in chapter 6.3.

Unfortunately due to technical issues and time constraints the simulations could not be used to validate the Mission Planner algorithm or its real-life operation via the Mission Operator. The HIL simulation could not be used because of an unsolved issue with the real-time performance. Solving the technical issues made it the interface to the non-real-time simulation was not fully completed.

The simulation environment would have provided valuable information before the real-life tests. Instead of a simulation, a simple Matlab script was used for the initial tests. A simple script was developed to run the algorithms from Matlab. With the script it was possible to evaluate if it is reasonable to expect the algorithms to be able to successfully complete a test drive in real-life. However, the script is only able to represent an ideal scenario where no real-life conditions are considered.

6.1 Non-real-time Simulation

A non-real-time simulation was implemented for algorithm testing and initial validation. The Simulink model for the simulation is shown in figure 24. An extrinsic wrapper function was used for the interface to the Mission Planner Matlab class. The extrinsic wrapper function contains a persistent instance of the Mission Planner class. The block named SIMULATOR contains the simulation of the vehicle kinematics

and navigation.

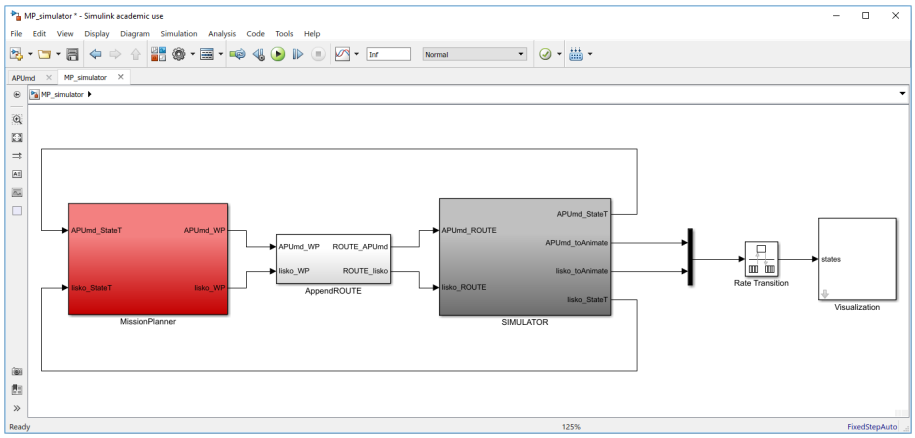


Figure 24: Simulink model for the non-real-time simulator.

6.2 Real-time HIL Simulation

A HIL simulation can be used to ease control system testing by replacing the process under control with a simulation. A HIL simulation allows system and device testing in an environment equivalent to real-life, removing various safety concerns and other challenges related to the actual process or physical system. In this thesis, the purpose of the HIL simulation was to allow initial testing of the entire system including the UDP interface without setting up the full real-life test set-up including the vehicles. Then the operation of the real-time guidance could be verified at the office before travelling to the test field and setting up the real-life test equipment.

Simulink Desktop Real-Time (SLDRT) was used to provide the real-time functionality for the HIL simulation. SLDRT provides a real-time kernel that allows Simulink normal mode to run in real-time. The Simulink model is synchronized with a real-time clock using SLRDT library blocks.

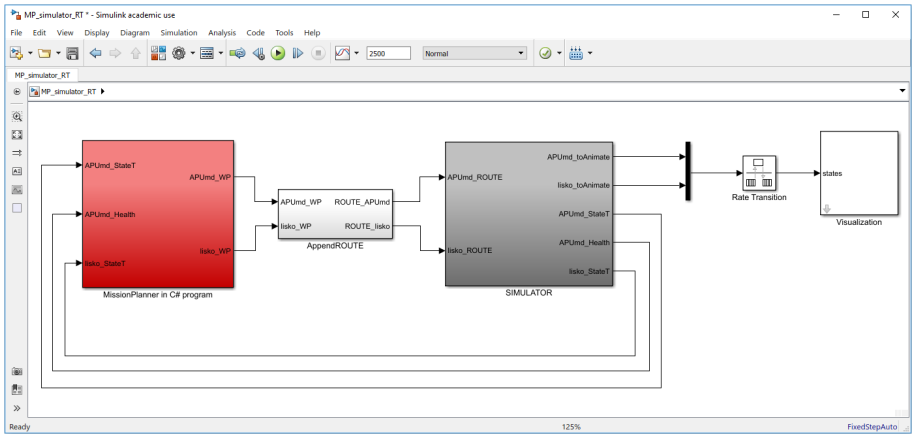


Figure 25: Packet output packet input UDP interface in the HIL simulation.

For the communication interface using UDP datagrams, the SLDRT blockset includes packet input and packet output blocks that support UDP data transfer. The simulation interface to the real-time guidance application was planned to be identical to that of the vehicles such that the same UDP messages can be used for both. This interface was fully implemented with the packet transfer blocks and is the main difference between the two simulations. The UDP block interface in the simulation is shown in figure 26.

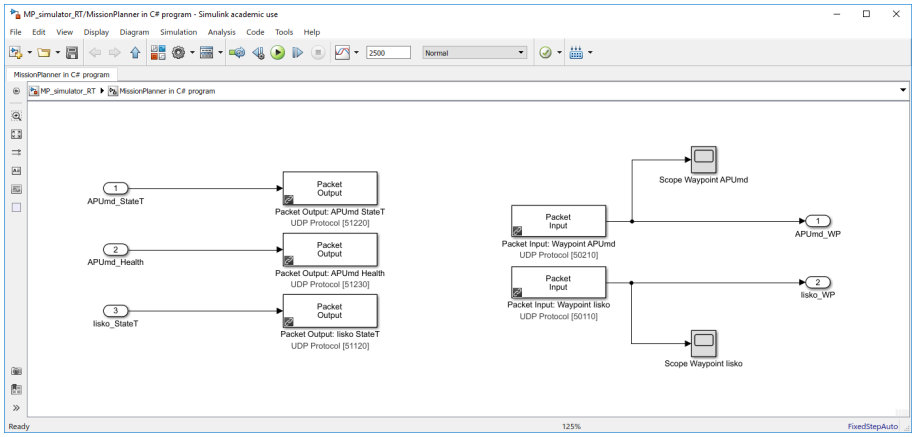


Figure 26: Simulink model for the real-time HIL simulator.

The real-time simulation was planned to run on a separate computer. The UDP messages would be sent over Ethernet between the simulation computer and the computer running the Mission Operator C# solution. The packet output UDP blocks send the (simulated) vehicle status messages to the Mission Operator C# solution and the packet input UDP blocks receive the waypoint messages from the Mission Operator C# solution. The connections between the simulation computer and the Mission Operator computer as they were planned are shown in figure 27. From the point of view of the Mission Operator computer the situation is equal to the one in the real-life test set-up.

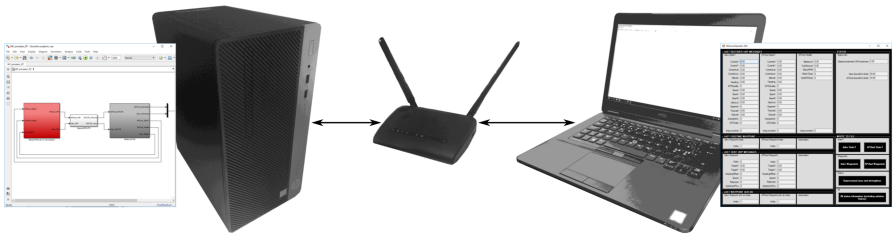


Figure 27: Real-time HIL simulation execution on two PCs.

6.3 Simulation Visualization

An animated visualization was implemented to be used with the simulations for visual validation of the algorithm operation. The animation was written as a level-1

S-function in Matlab code using Matlab figures. The visualization window includes the animation and some important status data of the vehicles. An example of the visualization is shown in figure 28 where the vehicles are driving a predetermined route inside the test field boundaries (in local coordinate frame).

The animation shows the field boundary and a simplified depiction of the vehicles that indicates their current location, heading and implement position. The implement position is indicated with color intensity of the vehicle fill based on hitch level data from the simulation. Hitch level 100.0 % corresponds to full intensity color and hitch level 0.0 % corresponds to white. Between these levels a linear mapping is used for the color. The field boundary is input as a shapefile and it is the same boundary that is used for cooperative route planning in the Mission Planner.

The visualization can be included in a simulation by using a level-1 S-function block. In addition to the typical inputs and outputs of a level-1 S-function, the following data is expected from the simulation: for both vehicles, current X and Y coordinate in YKJ, GPS heading, speed, hitch level, current waypoint, and estimated time and distance to the route end. Because continuously updating the animation figure demands a lot of processing time, a rate transition block was used between the simulation and the animation S-function block. The S-function block can be seen in the Simulink models in figures 24 and 25.

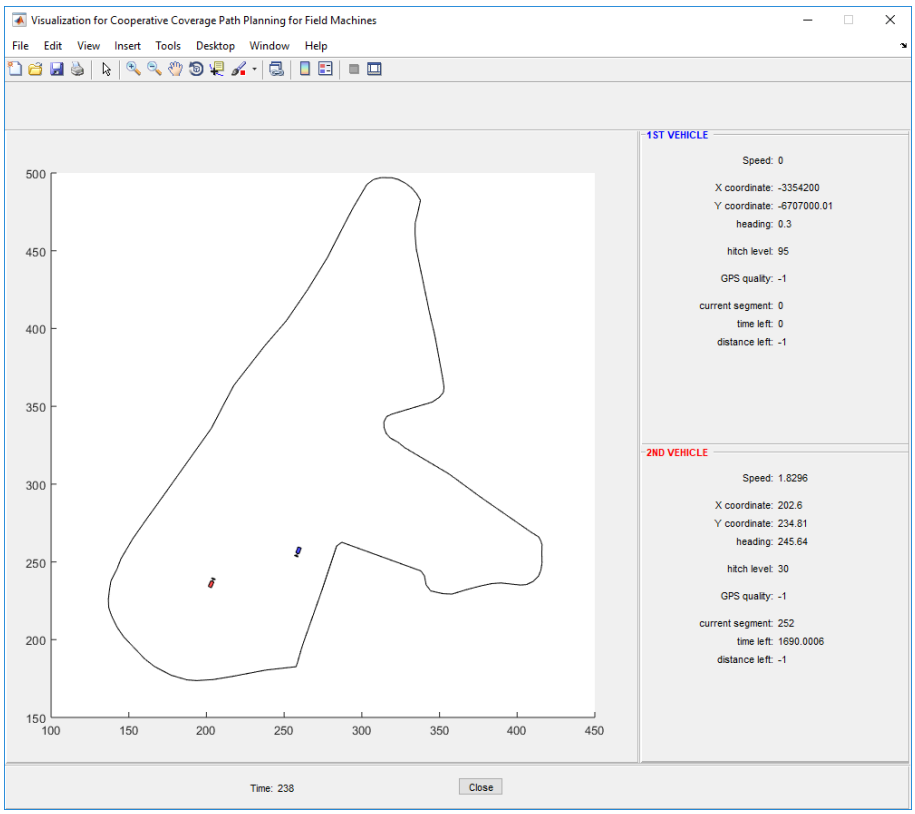


Figure 28: Simulation visualization.

7 Real-life Test Environment

In this study, we consider a scenario with two field machines performing different field operation tasks. The tasks are sequentially dependent, that is, the first one needs to be executed before the other. Thus the setting is co-operative while at the same time the units have individual tasks.

In the real-life test set-up constructed for this project two field machines were used. One is a fully autonomous four wheel drive tractor whereas the other is a semi-autonomous tractor that requires an assisting human operator. The latter is steered with front wheels only. The tractors have different tasks that need to be executed strictly sequentially.

Both tractors are equipped with necessary implements to perform the tasks. The semi-autonomous tractor is equipped with a disc cultivator and performs the first task. The autonomous tractor is equipped with a seeder and performs the second task.

7.1 Tractors and Implements

The cooperative real-life test scenario involves two tasks and two tractors. The first task is to cultivate the field with a disc cultivator and the second task is to sow the field with a seed drill. The first task is assigned to a semi-autonomous tractor with a driver. The second task is dependent on the first task and it is assigned to a fully autonomous driverless tractor.

Tractor with a Disc Cultivator

The tractor that performs the first task is a Valtra prototype. An image of the tractor is shown in figure 29. A human operator is always needed in the tractor. The tractor is otherwise able to follow a given path on its own but to change gear from forward to reverse or vice versa a lever needs to be operated manually. The navigation system requests the driver to change gear when necessary.

Figure 29: The tractor that performs the harrowing task.

The tractor is equipped with a hitch mounted DiscMaster 300 offset disk cultivator by Multiva. An image of the disc cultivator is shown in figure 30. The disc cultivator has a roller and two rows of cultivation discs. The cultivation depth can be controlled by adjusting the position of the roller. The disc cultivator weighs approximately 2000 kg and its working width is 3.0 meters. When the cultivator is in working position the tractor should not reverse.

Figure 30: The tractor that performs the cultivating task.

Tractor with a Seeder

The tractor that performs the second task (seeding) was originally built by a Finnish company called Modulaire Oy, but the control system was later reconditioned com-

pletely. The wheelbase is 2.7 m and the tractor weighs 5900 kg. Each wheel of the four wheel steered tractor steers maximum 22 degrees. The steering rate is limited to 8-12 degrees per second, and therefore the operating speed of the tractor in accurate navigation is limited to 2.0 meters per second. [93] The tractor is driverless and does not have a cabin. An image of the tractor is shown in figure 31.

Figure 31: The tractor that performs the seeding task.

The tractor is equipped with a hitch mounted KL 2500 seed drill by TUME. An image of the seed drill is shown in figure 32. The seed drill weighs approximately 1000 kg and its working width is 2.5 meters. When the seed drill is in working position the tractor should not reverse or make steep turns.

Figure 32: The tractor that performs the seeding task.

7.2 Hardware and Communication

The navigation systems and the communication hardware in the tractors are out of the scope of this thesis. It is sufficient to know that both vehicles have a navigation system computer on board and that they receive waypoint messages over UDP. The waypoints to the second vehicle are sent over a radio link. Both vehicles are equipped with RTK GPS receivers. The computer that runs the cooperative path planning algorithm in the Mission Operator software during the test drives is a Dell Latitude E5470 laptop with an Intel Core i5-6300U CPU rated at 2.4-2.5 GHz using 8 GB of RAM. The operating system is Windows 10 Pro. The data communication between the tractors and the Mission Operator computer are shown in figure 33. The Mission Operator computer is located in the tractor with the disc cultivator that performs the first task. The communication between the vehicles and the Mission Operator was implemented over Ethernet and UDP. As shown in figure 33, the physical connection is between the semi-autonomous tractor and the Mission Operator computer (1), and the vehicles are connected with a radio link (2).

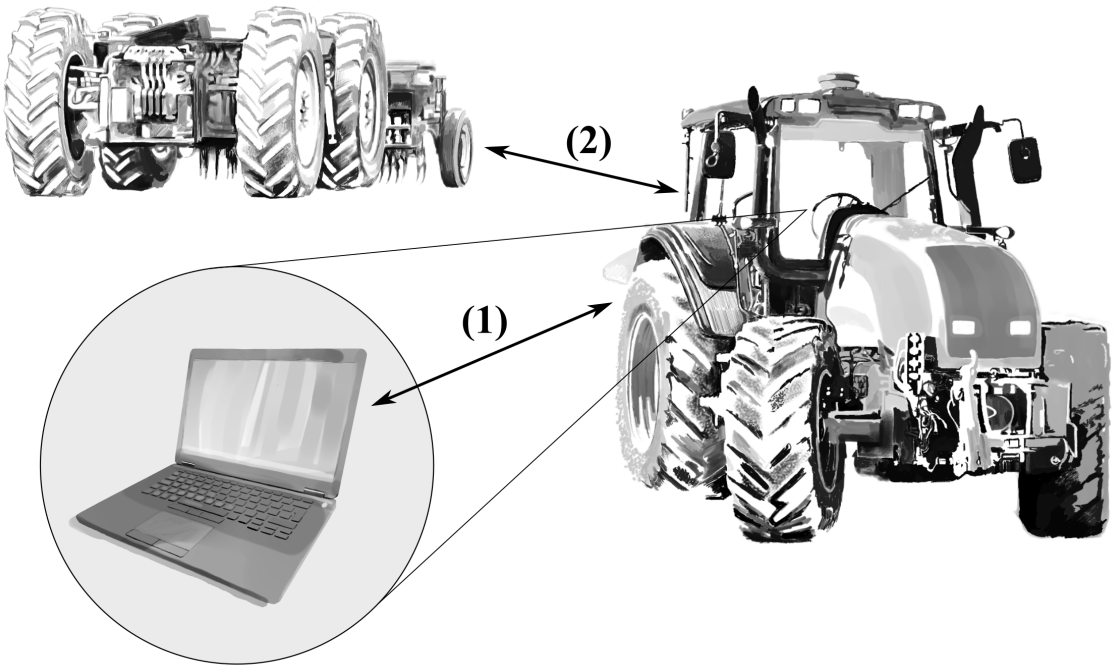


Figure 33: Communication in the real-life test environment: (1) Ethernet UDP between the Mission Operator and the semi-autonomous tractor, and (2) a radio link between the tractors.

7.3 Test Field Area

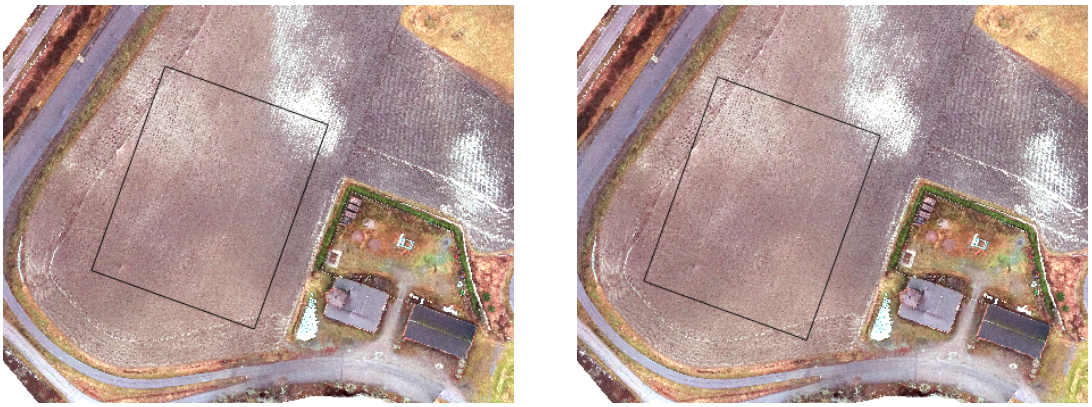
The test field that was used for the real-life tests during this study is located in Vihti, Finland. An aerial image of the test field area is shown in figure 34. There are minor

differences in altitude across the field. Several smaller test areas of different shapes and sizes were generated inside the field boundaries. Many of these test areas were mainly used for algorithm development.

Figure 34: An aerial image of the test field area in Vihti, Finland.

A rectangular area of 0.8 hectares (80 meters times 100 meters) was used for the final test drives. The final test drives were performed during September and October of year 2018. Due to changing field conditions during late autumn the test area was relocated once inside the field boundaries. After rain the water would gather in some parts of the field which made it difficult for the tractors to navigate autonomously without slipping.

Both test areas can be seen in figure 35. The size and shape of both test areas are the same and altitude differences are minor. White areas in the aerial images indicate the locations where the water tends to accumulate. In late September the test area 1 was still usable without problematic slipping conditions.



(a) Test area 1.

(b) Test area 2.

Figure 35: Two 0.8 hectare rectangular test areas for the final test drives.

8 Results

The algorithms were tested mainly in a real-life test environment with two tractors equipped with a disc cultivator and a seeder. It was originally planned that the algorithms would be evaluated in a simulation as well, but due to technical issues and time constraints this was eventually not possible. The real-life test results are presented in chapter 8.2.

8.1 Simulated Results

The implementation of a simulation environment was originally planned as a part of this thesis. Unfortunately due to technical issues and time constraints, the simulation could not be used to validate the algorithms. Initial evaluation of the algorithms was conducted with a Matlab script that is able to execute the algorithms in the Mission Planner class to produce a solution in an ideal case where no real-life constraints are considered. These results were only an initial validation and are not presented here since the algorithms were successfully tested in a real-life test environment. A Monte Carlo evaluation could be conducted with the test script but it is left for future work.

8.2 Real-life Test Results

Algorithm A21, Algorithm A31 and Algorithm B were tested in Vihti, Finland with the test set-up previously described in chapter 7. A detailed visual description of the progress of each test drive is presented in the Appendixes A and B. The dates and approximate durations of each test drive are shown in table 6. Algorithm A21 test drive was performed on test area 2 shown in figure 35b. Algorithm A31 and Algorithm B test drives were performed on test area 1 shown in figure 35a.

Table 6: Algorithm test drives in Vihti, Finland.

Algorithm	Date	Test drive duration
Algorithm A21	Mon 2018-10-15	02:09:00
Algorithm A31	Thu 2018-10-11	00:46:00
Algorithm B	Fri 2018-09-28	01:42:00

To provide a visual reference of the tests that were conducted in the field, an aerial image that was taken during the Algorithm A31 test drive is shown in figure 36. The photos were taken and compiled by Timo Oksanen.

Figure 36: An aerial image of Algorithm A31 test drive progress. Photo by Timo Oksanen.

Measured as a percentage of total area coverage, the first vehicle completed 99.99 % of its task and the second vehicle completed 93.42 % of its task during Algorithm A21 test drive. During Algorithm A31 test drive the first vehicle completed 98.36 % of its task and the second vehicle completed 55.22 % of its task. During Algorithm B test drive the first vehicle completed 100.00 % of its task and the second vehicle completed 96.25 % of its task. In Algorithm B test drive both vehicles drove all swaths. The completion percentages for the second vehicle would be higher if its parameters were tuned in more detail.

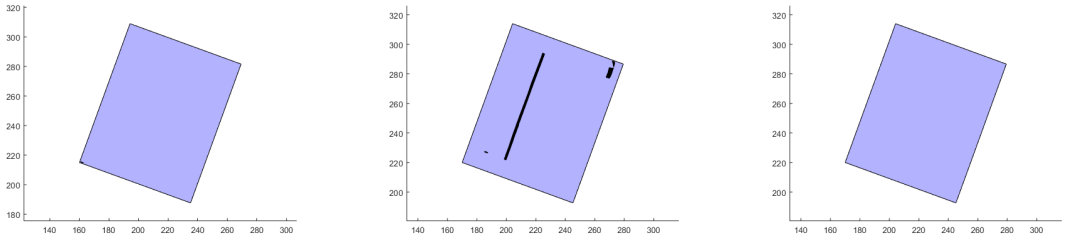
Algorithm A21 and Algorithm A31 were unable to fully complete the test drive. At the end of Algorithm A21 test drive, the second vehicle was unable to complete one swath. At the end of Algorithm A31 test drive, one swath remains for the first vehicle and twenty-one swaths remain for the second vehicle. Algorithm B was able to complete the test drive. When looking at the following data and statistics it should be considered that all values for different algorithms are not comparable due to these circumstances.

To evaluate the degree of completeness of the end results, the number of unfinished swaths at the end of each algorithm test drive are given in table 9. The total amount of swaths is equal for all algorithms since the swaths are computed in advance. For the first vehicle there are 16 swaths in the headland and 21 swaths in the mainland which is 37 swaths in total. For the second vehicle there are 42 swaths in total, 20 swaths in the headland and 22 swaths in the mainland.

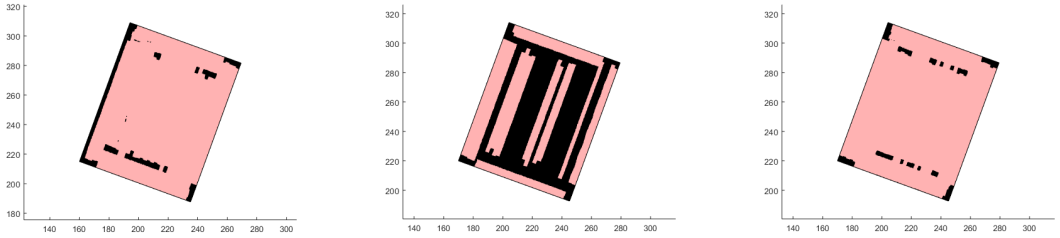
Table 7: Number of unfinished swaths at the end of each algorithm test drive.

Algorithm	1st vehicle	2nd vehicle
Algorithm A21	0	1
Algorithm A31	1	21
Algorithm B	0	0

In addition, the unprocessed area for all algorithm test drives for both vehicles at the end of each test drive is shown in black in figure 37. The completed area for the first vehicle is shown in blue and the completed area for the second vehicle is shown in red.



(a) Algorithm A21 1st vehicle. (b) Algorithm A31 1st vehicle. (c) Algorithm B 1st vehicle.



(d) Algorithm A21 2nd vehicle. (e) Algorithm A31 2nd vehicle. (f) Algorithm B 2nd vehicle.

Figure 37: Unprocessed area at the end of each algorithm test drive.

The corresponding numbers for absolute area and percentage of total absolute area for the end results shown in figure 37 are given in table 8. Total absolute area of both test areas is 0.8 ha.

Table 8: Unprocessed area at the end of each algorithm test drive.

Algorithm	1st vehicle	2nd vehicle
Algorithm A21	0.000047 ha (0.01 %)	0.052695 ha (6.58 %)
Algorithm A31	0.013085 ha (1.64 %)	0.398267 ha (49.78 %)
Algorithm B	0.000000 ha (0.00 %)	0.030010 ha (3.75 %)

To evaluate the amount of simultaneous cooperation for each algorithm, some statistics on the amount of simultaneous work during the algorithm progress are given in table 9 for each algorithm test drive. The total number of steps during the test drive is given in the first column and for each algorithm it corresponds to the number of subfigures in figures 40, 42 and 44 where at least either of the vehicles is working. In all columns the absolute number of steps is followed by the percentage of total number of steps in parentheses. For example, during Algorithm A21 test drive the vehicles were working simultaneously during 25 steps which is 48.1% of a total of 52 steps.

Again these numbers are not entirely comparable. It is typical for all the algorithms that the second vehicle has to wait in the beginning of the task and when the first

vehicle has completed the headland. The first vehicle waits in the end of the task. The steps where both vehicles wait are not included in the numbers. During Algorithm B test drive the first vehicle was manually driven outside the field boundaries after completing its task. From the algorithm point of view this counts as one step of the algorithm and it can be seen also in figure 44.

Table 9: Number of steps where the vehicles worked simultaneously for each algorithm test drive.

Algorithm	total	simultaneous	2nd vehicle waits	1st vehicle waits
Algorithm A21	52 (100.0 %)	25 (48.1 %)	11 (21.1 %)	16 (30.8 %)
Algorithm A31	36 (100.0 %)	20 (55.6 %)	16 (44.4 %)	0 (0.0 %)
Algorithm B	57 (100.0 %)	22 (38.6 %)	15 (26.3 %)	20 (35.1 %)

To evaluate the computational requirements of each algorithm, the computation times for each algorithm cycle were recorded during the test drives. One cycle of each algorithm corresponds to solving one turn and swath for both vehicles. The graph in figure 38 shows the computation times for all algorithm cycles for each algorithm test drive. The horizontal axis shows the algorithm cycles in chronological order and the vertical axis shows the cycle durations in seconds. Algorithm A21 is plotted with blue, Algorithm A31 is plotted with red and Algorithm B is plotted with black.

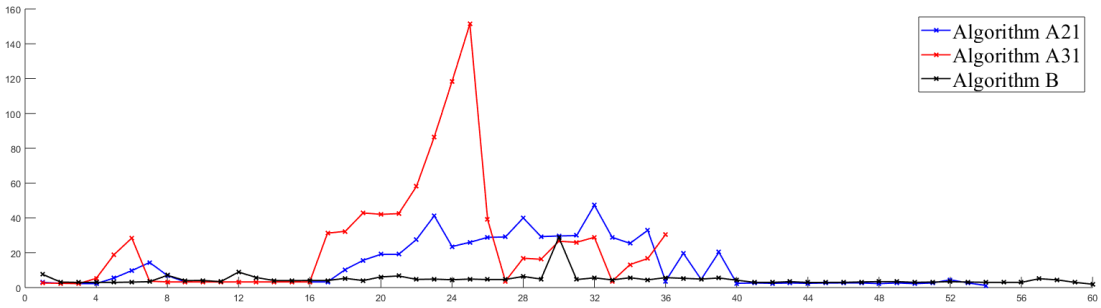


Figure 38: Computation times in seconds for each algorithm cycle of each algorithm test drive.

Statistics related to computation time are shown in table 10. The values correspond to the duration of one cycle of each algorithm. For example, the maximum time it took for Algorithm B to solve one simultaneous turn and swath for both vehicles was 28.86 seconds. Notice that this does not include the duration it took for the vehicles to complete the swaths. The total computation time during the test drives is shown in the last column. Notice that Algorithm A21 and Algorithm A31 were unable to complete the task and therefore the total time does not correspond to a full solution.

Table 10: Algorithm cycle computation times until the end of each algorithm test drive.

Algorithm	Minimum	Maximum	Average	Median	Total
Algorithm A21	1.22 sec	47.33 sec	12.34 sec	3.46 sec	666.46 sec
Algorithm A31	2.23 sec	151.34 sec	25.54 sec	16.45 sec	919.46 sec
Algorithm B	1.92 sec	28.86 sec	4.68 sec	4.05 sec	280.76 sec

These statistics should be seen as referential due to the uncertainties between the test drives. As the amount of optional swaths decrease towards the end of the task, the computation time during one cycle of the algorithm decreases as well. All of the statistics are not exactly comparable because Algorithm A21 and Algorithm A31 were unable to complete the test drive.

Of the three algorithms that were tested in real-life conditions, Algorithm A31 was able to solve the least amount of steps before the test drive was quit after step 36 due to a deadlock. To prepare a more comparable statistic on the algorithm cycle computation times we omit the later steps from Algorithm A21 and Algorithm B data so that we can compare steps 1-36 of solution. The algorithm cycle computation times until step 36 of each algorithm are shown in table 11.

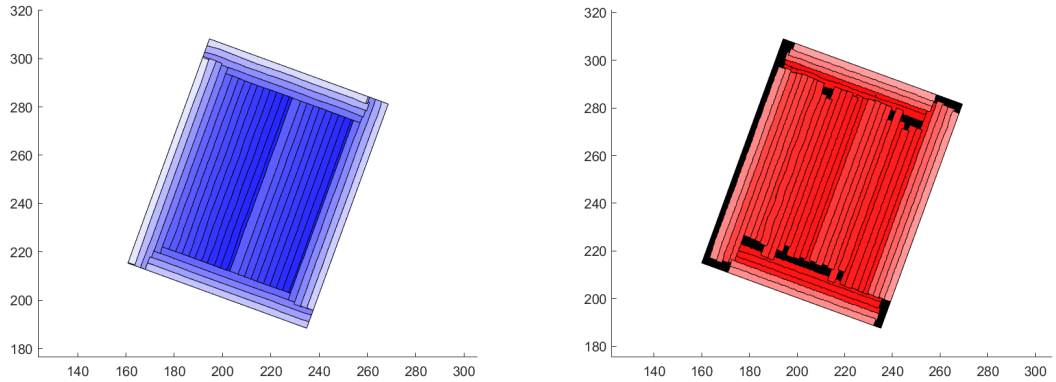
Table 11: Algorithm cycle computation times until round 36 of each algorithm test drive.

Algorithm	Minimum	Maximum	Average	Median	Total
Algorithm A21	2.23 sec	47.33 sec	16.20 sec	12.39 sec	583.16 sec
Algorithm A31	2.23 sec	151.34 sec	25.54 sec	16.45 sec	919.46 sec
Algorithm B	2.96 sec	28.86 sec	5.46 sec	4.62 sec	196.42 sec

Further visual details are given individually for each algorithm in the following sections. To see the step-by-step coverage result of each algorithm test drive the reader is referred to Appendixes A and B.

Algorithm A21 Test Drive

A test drive of Algorithm A21 was performed in Vihti, Finland on Mon the 15th of October 2018. Algorithm A21 is described in detail in chapter 5.3. The final coverage result for Algorithm A21 test drive is shown in figure 39. In the figures the 1st vehicle data is plotted with blue color and the 2nd vehicle data is plotted with red color. Color intensity indicates the temporal order of the swaths so that the most recent swaths are more intense in color. Black color indicates an area that was left uncovered. The coverage area is based on the vehicle GPS coordinates and the hitch level value which describes the vertical position of the tool. The coordinates in all figures are in local coordinate frame where y-axis corresponds to YKJ northing and x-axis corresponds to YKJ easting.



(a) Area covered by 1st vehicle.

(b) Area covered by 2nd vehicle.

Figure 39: Covered area at the end of Algorithm A21 test drive.

To demonstrate the progress of the test drive of Algorithm A21, all subsequent paths for both vehicles are shown in figure 40. One subfigure shows one turn and swath for both vehicles. If only the current position is shown for either vehicle it implies that the vehicle had to wait during that step. The steps are in chronological order from left to right, top to bottom so that the first step is shown in the top left subfigure.

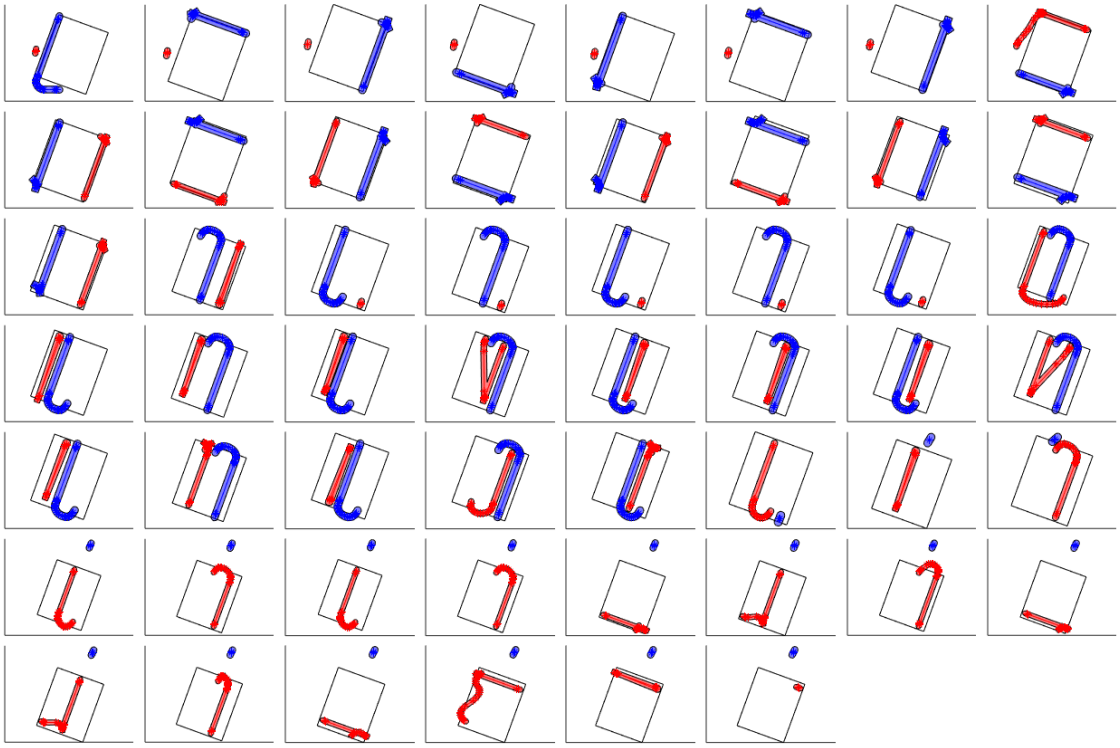
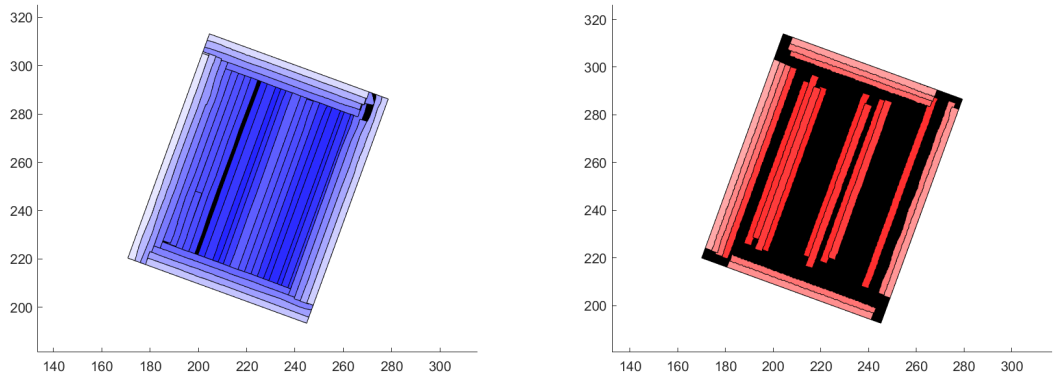


Figure 40: All simultaneous actions during Algorithm A21 test drive.

Algorithm A31 Test Drive

A test drive of Algorithm A31 was performed in Vihti, Finland on Thu the 11th of October 2018. Algorithm A31 is described in detail in chapter 5.3.

The final coverage result for Algorithm A31 test drive is shown in figure 41. In the figures the 1st vehicle data is plotted with blue color and the 2nd vehicle data is plotted with red color. Color intensity indicates the temporal order of the swaths so that the most recent swaths are more intense in color. Black color indicates an area that was left uncovered. The coverage area is based on the vehicle GPS coordinates and the hitch level value which describes the vertical position of the tool. The coordinates in all figures are in local coordinate frame where y-axis corresponds to YKJ northing and x-axis corresponds to YKJ easting.



(a) Area covered by 1st vehicle.

(b) Area covered by 2nd vehicle.

Figure 41: Covered area at the end of Algorithm A31 test drive.

To demonstrate the progress of the test drive of Algorithm A31, all subsequent paths for both vehicles are shown in figure 42. One subfigure shows one turn and swath for both vehicles. If only the current position is shown for either vehicle it implies that the vehicle had to wait during that step. The steps are in chronological order from left to right, top to bottom so that the first step is shown in the top left subfigure.

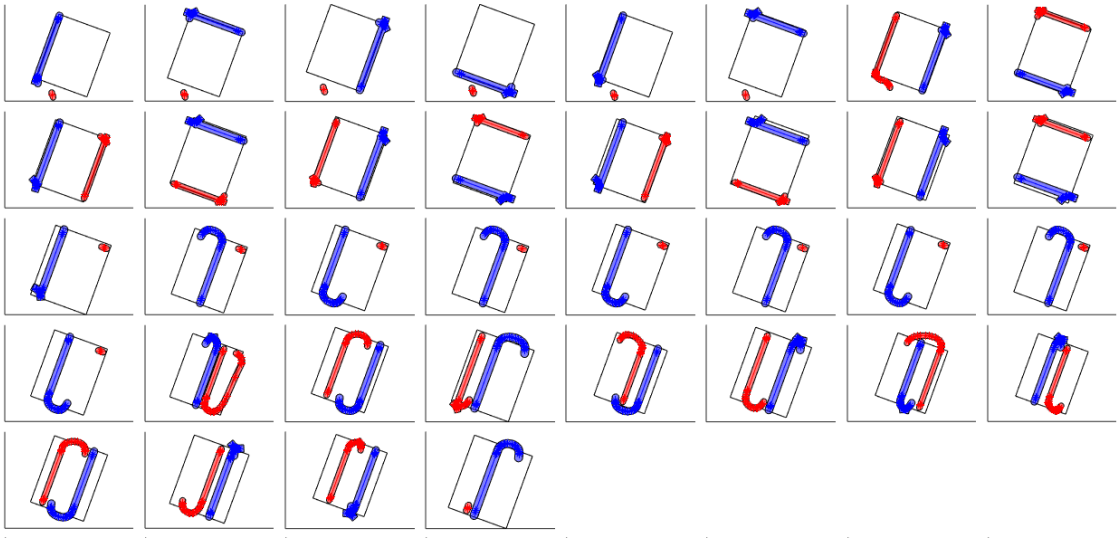
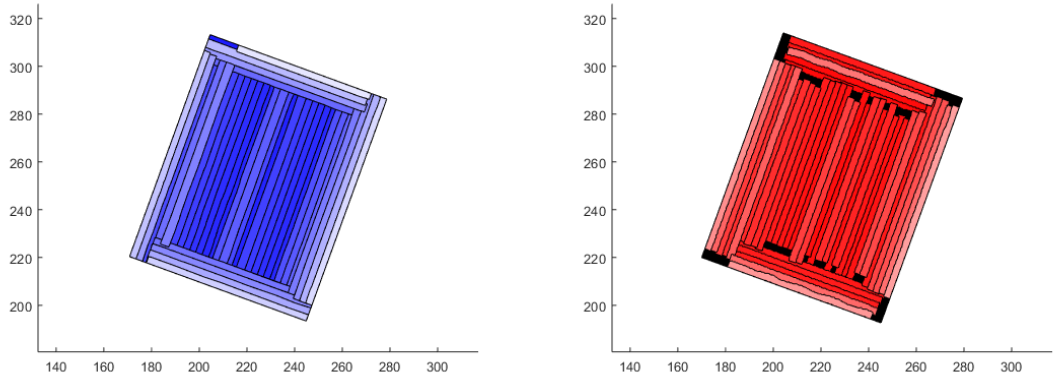


Figure 42: All simultaneous actions during Algorithm A31 test drive.

Algorithm B Test Drive

A test drive of Algorithm B was performed in Vihti, Finland on Fri the 28th of September 2018. Algorithm B is described in detail in chapter 5.3. The final coverage result is shown in figure 43. In the figures the 1st vehicle data is plotted with blue color and the 2nd vehicle data is plotted with red color. Color intensity indicates the temporal order of the swaths so that the most recent swaths are more intense in color. Black color indicates an area that was left uncovered. The coverage area is based on the vehicle GPS coordinates and the hitch level value which describes the vertical position of the tool. The coordinates in all figures are in local coordinate frame where y-axis corresponds to YKJ northing and x-axis corresponds to YKJ easting.



(a) Area covered by 1st vehicle.

(b) Area covered by 2nd vehicle.

Figure 43: Covered area at the end of Algorithm B test drive.

To demonstrate the progress of the test drive of Algorithm B, all subsequent

paths for both vehicles are shown in figure 44. One subfigure shows one turn and swath for both vehicles. If only the current position is shown for either vehicle it implies that the vehicle had to wait during that step. The steps are in chronological order from left to right, top to bottom so that the first step is shown in the top left subfigure.

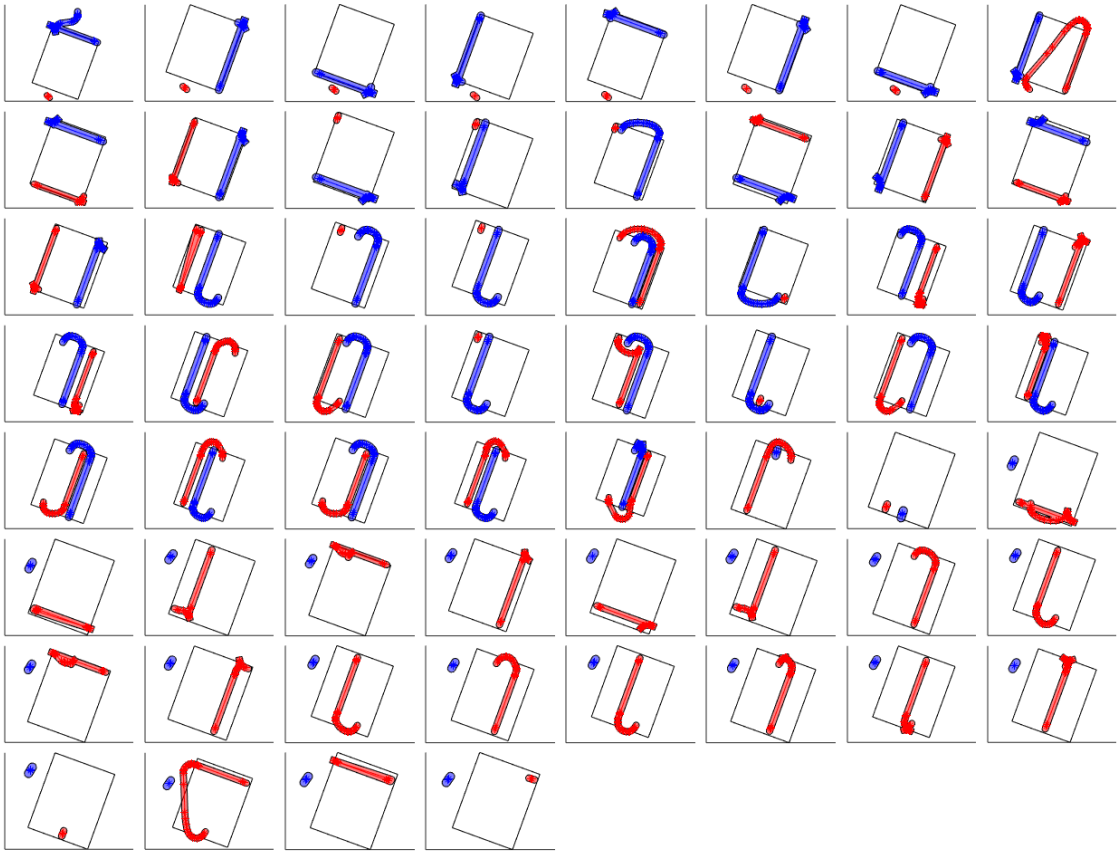


Figure 44: All simultaneous actions during Algorithm B test drive.

Test with a Restricted Turn Area

A separate test, without cooperation, was performed with a restricted turn area so that the tractor was allowed to cross the work area boundary only by a 1.5 meter margin. The test was performed to validate that the algorithms could solve the tasks even if the area around the field is not available for turning. The test was performed with the semi-autonomous tractor equipped with the disc cultivator. The result of the test drive is shown in figure 45, where the GPS data is shown in blue. The outer square is the turn area and the inner square is the work area. The turn area has the dimensions of 50 meters times 50 meters and the work area has the dimensions of 47 meters times 47 meters. It can be estimated that the vehicle stayed inside the turn area boundaries with reasonable accuracy.

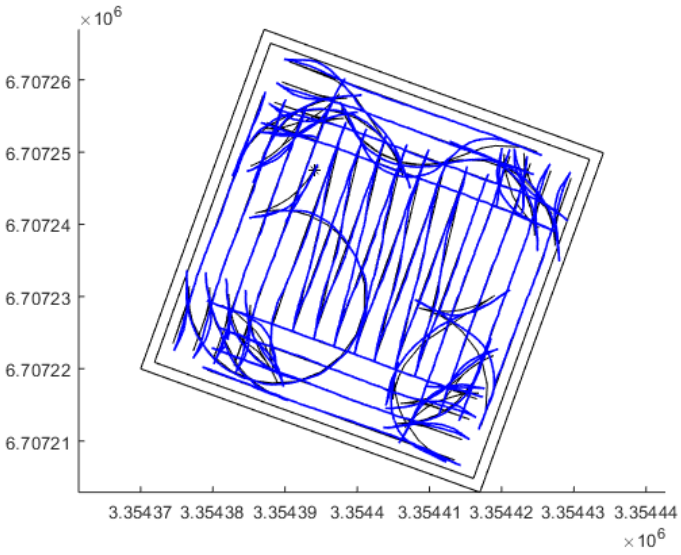


Figure 45: Planned path (black) and GPS data (blue) during the test with a restricted turn area.

9 Discussion

With the algorithms that were developed in this thesis it is possible to perform two sequentially dependent agricultural coverage tasks simultaneously on the same area. It was proven by tests in a real-life test environment with two tractors that were equipped with a disc cultivator and a seeder. A rectangular test area of 0.8 ha was used for the tests.

The results presented in chapter 8 give tentative indication that Algorithm B shows more reliable performance in solving the cooperative coverage path planning problem than Algorithm A. However, the fact that Algorithm A21 and Algorithm A31 did not fully complete the test drive is partly due to uncertainties and unfortunate errors during the test drives.

Both the test results and the test methods are assessed in the rest of this chapter. Some comparisons between the algorithms can be made based on the results. Also some general remarks about the algorithm behaviour are given based on what was observed during the test drives. Some suggestions for future improvements are given at the end of this chapter.

9.1 Assessment of the Test Results

Of the three test drive results presented in this study, Algorithm B showed the most reliable overall performance. In Algorithm B test drive both vehicles were able to complete all swaths. When measured as a percentage of total work area the first vehicle completed 100.00 % of its task and the second vehicle completed 96.25 % of its task.

On the other hand, despite the second vehicle did not finish the last swath during the test drive, Algorithm A21 showed the highest relative amount of truly simultaneous cooperation with a total of 25 actions being completed simultaneously. The reason that Algorithm B did not perform as well in this regard could be because the implement was not lowered in time during the first headland swath and therefore the corresponding swath for the second vehicle was not considered available until the end of the task when the incomplete swath was manually corrected by the driver of the first tractor. If Algorithm A31 were able to complete the test drive, it is likely that either 20 or 21 actions of a total of either 57 or 58 actions would have been truly simultaneous.

Algorithm A31 test drive was quit due to a deadlock. The second vehicle was unable to proceed while the first vehicle could not be rerouted due to having only one swath left. Only one swath was left for the first vehicle and 21 swaths for the second vehicle. A different reroute policy could have been able to continue the task. The fact that Algorithm A31 was unable to complete the test drive can be partly considered an unlucky event. Same goes for Algorithm A21, where the second vehicle was unable to complete one swath. This swath was never considered available because the first vehicle left a small gap in the westernmost corner of the field.

The test results on algorithm cycle computation times indicate that the computational performance of Algorithm B is obviously faster and more consistent over time

than Algorithm A in general. This result could also be deduced from the fact that during rerouting, Algorithm A might end up computing several turn paths, which is one of the more computationally demanding parts of the algorithms.

A value of 0.1 was set for parameter c in the efficiency calculation in equation 2 to prioritize the headland swaths for the first vehicle. Therefore it is typical for all algorithms to begin the task in the headland. It is common to follow this practice in tillage and sowing operations even if it is not necessary with GPS guided vehicles.

In the beginning of the task it is obvious that the second vehicle has to wait. The test drives indicate that the second vehicle has to wait also when the first vehicle has completed the headland swaths. The turn from one mainland swath to another intersects some of the headland swaths and therefore these headland swaths are not an option for the second vehicle while the first vehicle is working in the mainland. These headland swaths are likely to be the last ones that the second vehicle completes during the task.

It is expected that the field size and shape contribute to the results. With the approach that was presented in this thesis, the number of swaths depends on the field geometry, and obviously the number of swaths has an impact on the possibilities for proficient cooperation. For example, the safety margins for collision detection would (and should) prevent the vehicles form working on adjacent swaths. It is also intuitive that a small area is generally more demanding for cooperation.

Given the complexity of the task there are many uncertainties that can affect the end result, including the fact that the algorithms can end up in a deadlock where both vehicles are unable to proceed. The inability to solve a conflict is inherently an attribute of the algorithm, however, the likelihood of ending up in that conflict is dependent on the surrounding circumstances as well. Due to the complexity of the issue it is often difficult to say exactly which factors contributed to an unsolved conflict and how strongly.

Several things in the real-life circumstances could affect the completion of the coverage tasks. One crucial requirement for the cooperation is that the first vehicle is able to fully complete its task. Otherwise the coverage map that is used to update the available swaths for the second vehicle is incomplete and therefore the second vehicle will not be able to complete all of its swaths. For example navigation error, GPS noise and inconsistent delay in lowering or lifting the implement can result in small gaps in the computation of the coverage map. Also if GPS accuracy is temporarily too low, for example the RTK correction signal is lost, the navigation stops, which could affect the coverage result. Even if the navigation stops, a small gap may result in the coverage map. Usually this can be fixed by manually reversing the vehicle before continuing the autonomous navigation.

Also some uncertainties in the algorithms, including the option for arbitrary start positions for the vehicles, could affect the end result if a swath for the first vehicle is not completely covered because of this. The impact of this feature can be mitigated, however, by keeping the initial start positions of the vehicles the same between the test drives.

The parameters for the first vehicle were adjusted several times to ensure that the swaths are fully completed. The parameters for the second vehicle were not adjusted

in detail and therefore some gaps remain at the end of each swath. These gaps are visible for example in figures 39b, 41b and 43b. The completion percentages for the second vehicle would be higher if its parameters were tuned in more detail.

In addition, a separate test was performed to prove that the path planning algorithm and the turn path algorithm can provide a solution also when the turn area is more limited. This was demonstrated for one vehicle with a small test area. To test the cooperative algorithms with a more restricted turn area it would be advisable to use a larger test area than the 0.8 ha in this study since the vehicles would be required to turn inside the work area.

9.2 Assessment of the Test Methods

A tool chain that involves Matlab, Simulink and Visual Studio was adapted for the development and testing of the solution. All interfaces were planned and validated to be compatible so that the same algorithms could be evaluated both in a simulation and in a real-life test set-up.

Implementation of a simulation environment was originally a part of the goals of this thesis, however, it could not be completed in time due to technical issues and time constraints. The final tests of the algorithms were performed on field with real tractors and equipment. The simulation environment would have provided valuable information before the real-life tests. Instead of a simulation, a simple Matlab script was used for the initial tests.

A simple script was developed to run the algorithms from Matlab. With the script it was possible to evaluate if it is reasonable to expect the algorithms to be able to successfully complete a test drive in real-life. However, the script is only able to represent an ideal scenario where no real-life conditions are considered. All in all it is necessary, independent of the quality of the simulation, to perform tests in a real-life test environment. Unlike in many studies, the algorithms developed in this thesis were tested in a real-life test environment.

The real-life test is essential to prove the functionality of the algorithms. However it is prone to uncertainties that cannot be fully controlled between test runs and therefore it has its shortcomings when it comes to comparing different algorithms. Some of these uncertainties are also visible in the results of the test runs that were included in this study. For example, a gap remains in one of the headland swaths for the first vehicle at the end of Algorithm A31 test drive. This was due to the unfortunate field conditions that resulted in slipping and the vehicle was briefly unable to navigate autonomously. However, this also demonstrates that a simulation is not always enough to validate the algorithms for real-life purposes.

Conducting the real-life tests is also time consuming and therefore only a limited number of tests can be done in a reasonable time. For several reasons, including safety reasons, at least two persons are required to supervise the operation of the tractors during the test drives. Partly due to the time constraints a relatively small test area was chosen for the test drives. However, testing on a small area could prove to be more demanding since a large test area could provide more options and flexibility for cooperation.

In the real-life tests some minor errors could be caused by the driver in the first tractor, as well as some of them could be corrected manually. Then it should be noted, however, that the type of error that needed to be corrected manually could maybe not be solved independently by the algorithm.

It should be noted that the completion percent of each task was estimated based on calculations based on GPS data and geometry. This means that there could be minor disparity between reality and the computed coverage map. However, for example at the end of Algorithm B test drive, no major gaps were observed in the field that would not be recognizable in the coverage maps presented in chapter 8 as well.

A Monte Carlo evaluation of the algorithms could be done with the test script (as well as with a proper simulation). This would be beneficial for estimating the success rate of the algorithms with different initial conditions as well as for comparing the performance of the algorithms. For example, the impact of the vehicle start positions on the end result could be analysed by multiple test runs as well. Due to time constraints this evaluation is left for future work.

Overall more test runs of each algorithm would be beneficial to be able to compare the algorithms in more detail. However, the main objective of this thesis was to implement an algorithm that is provably capable of completing the task, which can be verified with the current results. A more comprehensive testing of the algorithms is left for future work as well.

9.3 Suggestions for Improvement

Several improvements can be recommended based on the test results and the experience that was obtained about the known shortcomings of the algorithms during testing on the field.

The adaptability of the algorithms could be improved if the swaths were computed online based on the updated coverage map. This could increase the options for cooperation especially for the second vehicle, when instead of finding the available swaths from a set of predetermined swaths, the swaths could be placed inside the currently available area. In addition, an error in the coverage of a swath could be corrected later, because all of the uncovered area would be considered when computing the swaths. This would, however, increase computation requirements online.

The computation times presented in chapter 8 indicate that all of the available processing time was not fully utilized and that there is room for more intensive online calculation (including optimization) during the tasks. For example, for the test areas used in this study, the time it takes for the tractors to complete one mainland swath is around 55 seconds. For most algorithm cycles of Algorithm B this would leave almost 50 seconds time before new waypoints need to be sent to the vehicles. These numbers should be considered estimates but they give a general idea of the time constraints and the available processing time.

Many of the conflicts that can result in a deadlock could be solved if it was possible to route one of the vehicles to wait outside the field boundaries. Or, for

example in a cultivation and seeding operation, it would also be possible to relocate the vehicle somewhere where both tasks have already been performed. The relocation procedure would require minor changes to the algorithms. The turn path function could be utilized to compute the path to relocate the vehicles. In addition, the reroute policies for Algorithm A could be modified so that a deadlock is avoided even when the policy cannot be followed in the end. For example see Algorithm A31 test drive in chapter 8.

Another way to better enable cooperation and full completion of the second task, small errors (gaps) in the coverage map of the first task could be overlooked when finding available swaths for the second vehicle. Currently only the swaths whose footprint is fully inside the area that is already processed by the first vehicle are considered available for the second vehicle. For example, a threshold could be defined so that when the availability of the swaths for the second vehicle is updated, a gap that is smaller than the value of the threshold is ignored. However in certain situations this could lead to the second vehicle completing a swath earlier than it would be necessary.

One major takeaway from the real-life tests is that in the beginning of the task, the second vehicle waits for the first vehicle to complete several swaths. It should be possible to alter the algorithms so that the second vehicle can begin its task earlier, especially when the width of the implement for the first vehicle is wider than for the second vehicle. However, this also depends on the amount of headland swaths for both vehicles. In general, for a convex area, if the implement of the second vehicle has a smaller footprint and the headland for the first vehicle is inside the headland of the second vehicle, it should be possible for the second vehicle to begin its task earlier.

Finally, an interesting but challenging topic for further research would be to attempt optimizing the actions of both vehicles jointly and multiple steps further to the future. Another interesting question is that is it possible to evaluate the likeliness that a certain situation will end up in an unsolved conflict later on.

Many suggestions for improvement can be made based on the test results and several observations about the algorithm behaviour on the field. Some of these ideas may pose a new research question and some of them are more of a concern for product development.

10 Conclusions

The main objective of this thesis was to implement an algorithm to solve the problem of sequentially dependent cooperative coverage path planning for two vehicles. Extensive research can be found from the literature on the general problem of coverage path planning and different applications. Based on the literature review that was conducted in this thesis, not much prior research or applications exist on sequentially dependent cooperative coverage path planning. Similar problem statements can be found from the research of persistent cooperative coverage path planning, however, they do not answer the research questions proposed in this study.

In this thesis, two algorithms were developed to solve the proposed cooperative coverage path planning problem. It was demonstrated by tests in a real-life test environment that with these algorithms it is possible to perform two sequentially dependent agricultural coverage tasks simultaneously on the same area. Some suggestions for improvement were made based on the test results and several observations about the algorithm behaviour on the field.

Successful cooperation was observed to be the sum of several factors. During development the algorithms were successfully tested with one vehicle on several convex areas of different shape. While the cooperative version of the algorithm was tested on a relatively small rectangular area, the results can be generalized to some extent. It is expected that at least the field size, the number of headland swaths, the width of the implements, the difference of the implement widths and the convexity of the field have an impact on the end results. With regards to the field size it is also expected that a narrow field would yield different results in terms of cooperation than a field with equal dimensions. The complexity of the field polygon, as long as it is convex, is not expected to have a significant impact on the results. Any conclusions for a non-convex field should be done with caution. The number of headland swaths and the width of the implements for both vehicles affect the point where the second vehicle can theoretically begin its task. Both of these factors contribute to the size of the headland area. With the configuration of the headland that was used in this study it is advisable that the headland area for the second vehicle should be wider than for the first vehicle. Testing on a larger test area is assumed to provide more options and flexibility for cooperation.

While the algorithms implemented as a part of this thesis are able to solve the proposed coverage path planning problem, many new ideas arise for future development. Further research would enable generalizing the results for various real-life scenarios including concave field areas. This development is left for future work.

References

[1] United Nations Department of Economic and Social Affairs Population Division (2017). World population prospects: The 2017 revision, key findings and advance tables. https://esa.un.org/unpd/wpp/Publications/Files/WPP2017_KeyFindings.pdf. Working Paper No. ESA/P/WP/248. Online: accessed 2018-03-07.

[2] The Food and Agriculture Organization. Arable land (hectares per person). https://data.worldbank.org/indicator/AG.LND.ARBL.HA.PC. Online: accessed 2018-03-08.

[3] The Food and Agriculture Organization. Cereal yield (kg per hectare). https://data.worldbank.org/indicator/AG.YLD.CREL.KG. Online: accessed 2018-03-08.

[4] Denny Oetomo, John Billingsley, and John Reid. Editorial: Agricultural robotics. *Journal of Field Robotics*, 26:501–503, 2009.

[5] Myer Kutz. *Farm Machinery Automation for Tillage, Planting Cultivation, and Harvesting*, chapter 4. Elsevier, 2013.

[6] Esther M. Arkin, Sándor P. Fekete, and Joseph S.B. Mitchell. Approximation algorithms for lawn mowing and milling. *Computational Geometry*, 17(1):25 – 50, 2000.

[7] F. Yasutomi, M. Yamada, and K. Tsukamoto. Cleaning robot control. In *Proceedings. 1988 IEEE International Conference on Robotics and Automation*, pages 1839–1841 vol.3, April 1988.

[8] Prasad N. Atkar, Aaron Greenfield, David C. Conner, Howie Choset, and Alfred A. Rizzi. Uniform coverage of automotive surface patches. *The International Journal of Robotics Research*, 24(11):883–898, 2005.

[9] Martin Held. A geometry-based investigation of the tool path generation for zigzag pocket machining. *The Visual Computer*, 7(5):296–308, Sep 1991.

[10] Donghong Ding, Zengxi Pan, Dominic Cuiuri, Huijun Li, and Stephen van Duin. *Advanced Design for Additive Manufacturing: 3D Slicing and 2D Path Planning*, chapter 1. InTech, July 2016.

[11] Marina Torres, David A. Pelta, José L. Verdegay, and Juan C. Torres. Coverage path planning with unmanned aerial vehicles for 3D terrain reconstruction. *Expert Systems with Applications*, 55:441–451, 2016.

[12] Ercan U. Acar, Howie Choset, Yangang Zhang, and Mark Schervish. Path planning for robotic demining: Robust sensor-based coverage of unstructured environments and probabilistic methods. *The International Journal of Robotics Research*, 22(7-8):441–466, 2003.

[13] Cheng Fang and S. Anstee. Coverage path planning for harbour seabed surveys using an autonomous underwater vehicle. In *OCEANS 2010 IEEE - Sydney*, pages 1–8, May 2010.

[14] Andreas Bircher, Mina Kamel, Kostas Alexis, Michael Burri, Philipp Oettershagen, Sammy Omari, Thomas Mantel, and Roland Siegwart. Three-dimensional coverage path planning via viewpoint resampling and tour optimization for aerial robots. *Autonomous Robots*, 40(6):1059–1078, Aug 2016.

[15] M. Ollis and A. Stentz. First results in vision-based crop line tracking. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 1, pages 951–956, Apr 1996.

[16] Thomas Pilarski, Michael Happold, Henning Pangels, Mark Ollis, Kerien Fitzpatrick, and Anthony Stentz. The demeter system for automated harvesting. *Autonomous Robots*, 13(1):9–20, Jul 2002.

[17] Noboru Noguchi, Michio Kise, Kazunobu Ishii, and Hideo Terao. Coverage path planning with realtime replanning for inspection of 3D underwater structures. In *Automation Technology for Off-Road Equipment, Proceedings of the July 26-27, 2002 Conference*, pages 239–245, Chicago, Illinois, USA, May 2002.

[18] Timo Oksanen and Arto Visala. Coverage path planning algorithms for agricultural field machines. *Journal of Field Robotics*, 26(8):651–668, 2009.

[19] Yu-Song Jiao, Xin-Min Wang, Hai Chen, and Yan Li. Research on the coverage path planning of uavs for polygon areas. In *2010 5th IEEE Conference on Industrial Electronics and Applications*, pages 1467–1472, June 2010.

[20] E. Galceran, R. Campos, N. Palomeras, M. Carreras, and P. Ridao. Coverage path planning with realtime replanning for inspection of 3D underwater structures. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6586–6591, May 2014.

[21] Jian Jin and Lie Tang. Coverage path planning on three-dimensional terrain for arable farming. *Journal of Field Robotics*, 28(3):424–440, 2011.

[22] Peng Cheng, J. Keller, and V. Kumar. Time-optimal uav trajectory planning for 3D urban structure coverage. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2750–2757, Sept 2008.

[23] S. Fabre, P. Souères, M. Taïx, and L. Cordesses. Farmwork path planning for field coverage with minimum overlapping. In *ETFA 2001. 8th International Conference on Emerging Technologies and Factory Automation. Proceedings (Cat. No.01TH8597)*, volume 2, pages 691–694, Oct 2001.

[24] D.D. Bochtis and S.G. Vougioukas. Minimising the non-working distance travelled by machines operating in a headland field pattern. *Biosystems Engineering*, 101(1):1–12, 2008.

[25] Dionysis D. Bochtis. *Planning and Control of a Fleet of Agricultural Machines for Optimal Management of Field Operations*. PhD thesis, Aristitle University of Thessaloniki, School of Agriculture, Thessaloniki, Greece, 2008.

[26] Dionysis D. Bochtis, Claus G. Sørensen, Patrizia Busato, and Remigio Berruto. Benefits from optimal route planning based on b-patterns. *Biosystems Engineering*, 115(4):389–395, 2013.

[27] Timo Oksanen. *Path planning algorithms for agricultural field machines*. PhD thesis, Helsinki University of Technology, Espoo, Dec 2007.

[28] Sytze de Bruin, Peter Lerink, Aad Klompe, Tamme van der Wal, and Sanne Heijting. Spatial optimisation of cropped swaths and field margins using gis. *Computers and Electronics in Agriculture*, 68(2):185–190, 2009.

[29] J. Jin and L. Tang. Optimal coverage path planning for arable farming on 2D surfaces. *Transactions of the ASABE*, 53:283–295, 01 2010.

[30] Ibrahim A. Hameed, Dionysis Bochtis, and Claus A. Sørensen. An optimized field coverage planning approach for navigation of agricultural robots in fields involving obstacle areas. *International Journal of Advanced Robotic Systems*, 10(5), 2013.

[31] I.A. Hameed, A. la Cour-Harbo, and O.L. Osen. Side-to-side 3D coverage path planning approach for agricultural robots to minimize skip/overlap areas between swaths. *Robotics and Autonomous Systems*, 76:36–45, 2016.

[32] X. Yu, T. A. Roppel, and J. Y. Hung. An optimization approach for planning robotic field coverage. In *IECON 2015 - 41st Annual Conference of the IEEE Industrial Electronics Society*, pages 4032–4039, Nov 2015.

[33] X. Zhang, M. Geimer, L. Grandl, and B. Kammerbauer. Method for an electronic controlled platooning system of agricultural vehicles. In *2009 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pages 156–161, Nov 2009.

[34] Martin Andreas Falk Jensen, Dionysis Bochtis, Claus Grøn Sørensen, Morten Rufus Blas, and Kasper Lundberg Lykkegaard. In-field and inter-field path planning for agricultural transport units. *Computers & Industrial Engineering*, 63(4):1054–1061, 2012.

[35] J. Conesa-Muñoz, J. M. Bengochea-Guevara, D. Andujar, and A. Ribeiro. Efficient distribution of a fleet of heterogeneous vehicles in agriculture: A practical approach to multi-path planning. In *2015 IEEE International Conference on Autonomous Robot Systems and Competitions*, pages 56–61, April 2015.

[36] SAE International. J3016 surface vehicle recommended practice. Recommended practice, SAE International, September 2016.

[37] SAE International. Automated driving. https://www.sae.org/misc/pdfs/automated_driving.pdf. Online: accessed 2017-11-23.

[38] T. Chosa, M. Omine, and K. Itani. Dynamic performance of global positioning system velocity sensor for extremely accurate positioning. *Biosystems Engineering*, 97(1):3–9, 2007.

[39] Ahmed El-Rabbany. *Introduction to GPS: The Global Positioning System*. Artech Housel, Boston, 2002.

[40] Markku Poutanen. *GPS-paikanmääritys*. Tähtitieteellinen yhdistys Ursa, Helsinki, 1998.

[41] Navigation the National Coordination Office for Space-Based Positioning and Timing. Official u.s. government information about the global positioning system (gps) and related topics. https://www.gps.gov/systems/gps/modernization/civilsignals/, 2018. Online: accessed 2018-11-18.

[42] Richard B. Langley. Rtk gps. *GPS World*, 9(9):70–76, September 1998.

[43] Statistics Finland. Concepts: Map coordinate. http://www.stat.fi/meta/kas/koordinaatti_en.html. Online: accessed 2017-10-11.

[44] Statistics Finland. Concepts: Uniform coordinate. http://www.stat.fi/meta/kas/yhtenaiskoordin_en.htm. Online: accessed 2017-10-11.

[45] G.M. Ziegler. *Lectures on Polytopes*. Graduate Texts in Mathematics. Springer-Verlag, New York, 1995.

[46] Reinhard Diestel. *Graph Theory*. Graduate Texts in Mathematics. Springer-Verlag, Berlin, 3 edition, 2005.

[47] Abraham P. Punnen. *The Traveling Salesman Problem and Its Variations*, volume 12 of *Combinatorial Optimization Ser.* Springer-Verlag, Boston, MA, 2002.

[48] S. P. Anbuudayasankar, K. Ganesh, and Sanjay Mohapatra. *Models for Practical Routing Problems in Logistics: Design and Practices*. Springer International Publishing, Switzerland, 1 edition, 2014.

[49] Bruno Siciliano and Oussama Khatib. *Springer Handbook of Robotics*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2 edition, 2016.

[50] Alessandro Gasparetto, Paolo Boscariol, Albano Lanzutti, and Renato Vidoni. Path planning and trajectory planning algorithms: A general overview. In *Motion and Operation Planning of Robotic Systems*, volume 29, pages 3–27. Springer, Cham, 03 2015.

[51] Jean-Claude Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Norwell, MA, USA, 5 edition, 1998.

[52] Roland Siegwart, Illah R. Nourbakhsh, and Davide Scaramuzza. *Introduction to Autonomous Mobile Robots*. The MIT Press, Cambridge, MA, USA, 2 edition, 2011.

[53] Zuo Llang Cao, Yuyu Huang, and Ernest Hall. Region filling operations with random obstacle avoidance for mobile robots. *Journal of Robotic Systems*, 5:87–102, 04 1988.

[54] Howie Choset. Coverage for robotics – a survey of recent results. *Annals of Mathematics and Artificial Intelligence*, 31(1-4):113–126, may 2001.

[55] A. Zelinsky, R.A. Jarvis, J. C. Byrne, and S. Yuta. Planning paths of complete coverage of an unstructured environment by a mobile robot. In *In Proceedings of International Conference on Advanced Robotics*, pages 533–538, 1993.

[56] Wesley H. Huang. Optimal line-sweep-based decompositions for coverage algorithms. In *Proceedings - IEEE International Conference on Robotics and Automation*, volume 1, pages 27–32, 02 2001.

[57] Howie Choset and Philippe Pignon. Coverage path planning: The boustrophedon decomposition. In *Proceedings of the International Conference on Field and Service Robotics*, December 1997.

[58] Yoav Gabriely and Elon Rimon. Competitive on-line coverage of grid environments by a mobile robot. *Computational Geometry*, 24(3):197–224, 2003.

[59] Ercan U. Acar, Howie Choset, Alfred A. Rizzi, Prasad N. Atkar, and Douglas Hull. Morse decompositions for coverage tasks. *The International Journal of Robotics Research*, 21(4):331–344, 2002.

[60] Ercan U. Acar and Howie Choset. Sensor-based coverage of unknown environments: Incremental construction of morse decompositions. *The International Journal of Robotics Research*, 21(4):345–366, 2002.

[61] E. Garcia and P. Gonzalez de Santos. Mobile-robot navigation with complete coverage of unstructured environments. *Robotics and Autonomous Systems*, 46(4):195–204, 2004.

[62] A. E. F. Ryerson and Q. Zhang. Vehicle path planning for complete field coverage using genetic algorithms. *Agricultural Engineering International: the CIGR Ejournal*, IX, 08 2007.

[63] Tae-Kyeong Lee, Sanghoon Baek, Young-Ho Choi, and Se-Young Oh. Smooth coverage path planning and control of mobile robots based on high-resolution grid map representation. *Robotics and Autonomous Systems*, 59:801–812, 10 2011.

[64] Yan Li, Hai Chen, Meng Joo Er, and Xinmin Wang. Coverage path planning for uavs based on enhanced exact cellular decomposition method. *Mechatronics*, 21(5):876–885, 2011. Special Issue on Development of Autonomous Unmanned Aerial Vehicles.

[65] J. I. Vasquez-Gomez, M. M. Melchor, and J. C. H. Lozada. Optimal coverage path planning based on the rotating calipers algorithm. In *2017 International Conference on Mechatronics, Electronics and Automotive Engineering (ICMEAE)*, pages 140–144, Nov 2017.

[66] S. Bochkarev and S. L. Smith. On minimizing turns in robot coverage path planning. In *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 1237–1242, Aug 2016.

[67] Hoang Huu Viet, Viet-Hung Dang, Md Nasir Uddin Laskar, and TaeChoong Chung. Ba*: an online complete coverage algorithm for cleaning robots. *Applied Intelligence*, 39(2):217–235, Sep 2013.

[68] Junnan Song and Shalabh Gupta. ε^*: An online coverage path planning algorithm. *IEEE Transactions on Robotics*, 34:526–533, 02 2018.

[69] D. Kurabayashi, J. Ota, T. Arai, and E. Yoshida. Cooperative sweeping by multiple mobile robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 2, pages 1744–1749, Apr 1996.

[70] Chaomin Luo, S. X. Yang, and D. A. Stacey. Real-time path planning with deadlock avoidance of multiple cleaning robots. In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, volume 3, pages 4080–4085, Sept 2003.

[71] Simon Yang and Chaomin Luo. A neural network approach to complete coverage path planning. *IEEE transactions on systems, man, and cybernetics. Part B, Cybernetics : a publication of the IEEE Systems, Man, and Cybernetics Society*, 34:718–724, 03 2004.

[72] A. Ahmadzadeh, A. Jadbabaie, V. Kumar, and G. J. Pappas. Cooperative coverage using receding horizon control. In *2007 European Control Conference (ECC)*, pages 2466–2470, July 2007.

[73] Ioannis Rekleitis, Ai Peng New, Edward Samuel Rankin, and Howie Choset. Efficient boustrophedon multi-robot coverage: an algorithmic approach. *Annals of Mathematics and Artificial Intelligence*, 52(2):109–142, Apr 2008.

[74] Jose Palacios-Gasos, Zeynab Talebpour, Eduardo Montijano, C Sagues, and A Martinoli. Optimal path planning and coverage control for multi-robot persistent coverage in environments with obstacles. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1321–1327, 05 2017.

[75] J. M. Palacios-Gasós, E. Montijano, C. Sagüés, and S. Llorente. Cooperative periodic coverage with collision avoidance. *IEEE Transactions on Control Systems Technology*, pages 1–12, 2018.

[76] Carlos Franco, Dušan M. Stipanović, Gonzalo López-Nicolás, Carlos Sagüés, and Sergio Llorente. Persistent coverage control for a team of agents with collision avoidance. *European Journal of Control*, 22(Complete):30–45, 2015.

[77] Y. Jin, Y. Wu, and N. Fan. Research on distributed cooperative control of swarm uavs for persistent coverage. In *Proceedings of the 33rd Chinese Control Conference*, pages 1162–1167, July 2014.

[78] Nare Karapetyan, Kelly Benson, Chris McKinney, Perouz Taslakian, and Ioannis Rekleitis. Efficient multi-robot coverage of a known environment. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1846–1852, 09 2017.

[79] D. Zhu, C. Tian, X. Jiang, and C. Luo. Multi-auvs cooperative complete coverage path planning based on gbnn algorithm. In *2017 29th Chinese Control And Decision Conference (CCDC)*, pages 6761–6766, May 2017.

[80] Sina Sharif Mansouri, Christoforos Kanellakis, Emil Fresk, Dariusz Kominiak, and George Nikolakopoulos. Cooperative coverage path planning for visual inspection. *Control Engineering Practice*, 74:118–131, 2018.

[81] Dionysis Bochtis and T Oksanen. Combined coverage and path planning for field operations. In *Precision Agriculture 2009 - Papers Presented at the 7th European Conference on Precision Agriculture, ECPA 2009*, pages 521–527, Jul 2009.

[82] Mark Spekken and Sytze de Bruin. Optimized routing on agricultural fields by minimizing maneuvering and servicing time. *Precision Agriculture*, 14(2):224–244, Apr 2013.

[83] D. D. Bochtis and C. G. Sørensen. The vehicle routing problem in field logistics part i. *Biosystems Engineering*, 104(4):447–457, 2009.

[84] D. D. Bochtis and C. G. Sørensen. The vehicle routing problem in field logistics: Part ii. *Biosystems Engineering*, 105(2):180–188, 2010.

[85] Mernout Burger, Marco Huiskamp, and Tamás Keviczky. Complete field coverage as a multi-vehicle routing problem. *IFAC Proceedings Volumes*, 46(18):97–102, 2013. 4th IFAC Conference on Modelling and Control in Agriculture, Horticulture and Post Harvest Industry.

[86] Jesus Conesa-Muñoz, Gonzalo Pajares, and Angela Ribeiro. Mix-opt: A new route operator for optimal coverage path planning for a fleet in an agricultural environment. *Expert Systems with Applications*, 54:364–378, Jul 2016.

[87] T. Blender, T. Buchner, B. Fernandez, B. Pichlmaier, and C. Schlegel. Managing a mobile agricultural robot swarm for a seeding task. In *IECON 2016 - 42nd Annual Conference of the IEEE Industrial Electronics Society*, pages 6879–6886, Oct 2016.

[88] Hasan Seyyedhasani and Joseph S. Dvorak. Using the vehicle routing problem to reduce field completion times with multiple machines. *Computers and Electronics in Agriculture*, 134:142–150, 2017.

[89] Stavros G. Vougioukas. A distributed control framework for motion coordination of teams of autonomous agricultural vehicles. *Biosystems Engineering*, 113(3):284–297, 2012.

[90] Angus Johnson. Clipper - an open source freeware library for clipping and offsetting lines and polygons, 2010–2014. An open source freeware library for clipping and offsetting lines and polygons.

[91] The Internet Engineering Task Force. Internet standard rfc 768: User datagram protocol. Internet standard, The Internet Engineering Task Force, August 1980. Online: accessed 2017-11-30.

[92] J. F. Kurose and K. W. Ross. *Computer Networking: A Top-Down Approach*. Pearson, 2013.

[93] Timo Oksanen. *Accuracy and Performance Experiences of Four Wheel Steered Autonomous Agricultural Tractor in Sowing Operation*, pages 425–438. Springer International Publishing, Cham, 2015.

A Appendix

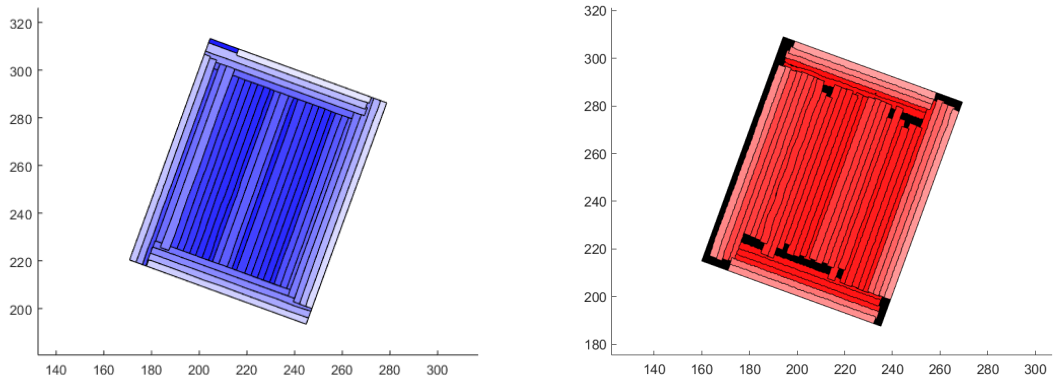
The details of two Algorithm A test drives in Vihti, Finland are presented in the following figures. Algorithm A21 was tested on Mon the 15th of October and Algorithm A31 was tested on Thu the 11th of October 2018. Number 21 indicates a reroute policy where the second vehicle is rerouted in case of a conflict. Number 31 indicates a reroute policy where alternate rerouting is used in a conflict situation. The only difference between Algorithm A21 and Algorithm A31 is the reroute policy. Algorithm A and the different reroute policies are described in detail in chapter 5.3.

Algorithm A21 Test Drive

In the figures the 1st vehicle data is plotted with blue color and the 2nd vehicle data is plotted with red color. The coordinates in all figures are in local coordinate frame where y-axis corresponds to YKJ northing and x-axis corresponds to YKJ easting.

The final coverage result is presented in figure A1. Color intensity indicates the temporal order of the swaths so that the most recent swaths are more intense in color. Black color indicates an area that was left uncovered. The coverage area is based on the vehicle GPS coordinates and the hitch level value which describes the vertical position of the tool.

The step by step progress of area coverage is presented in figures A2, A3, A4 and A5. In chronological order, each step describes the result of one cycle of the algorithm and shows the previously covered area for both vehicles. A marker indicates the initial location of the vehicle during the step. A continuous line indicates the turn and a coloured polygonal region indicates the coverage area of the swath. If the vehicle does not move during the step only the marker is shown. The coverage area is based on vehicle GPS coordinates and the hitch level value which describes the vertical position of the tool.



(a) Area covered by 1st vehicle. (b) Area covered by 2nd vehicle.

Figure A1: Covered area at the end of Algorithm A21 test drive.

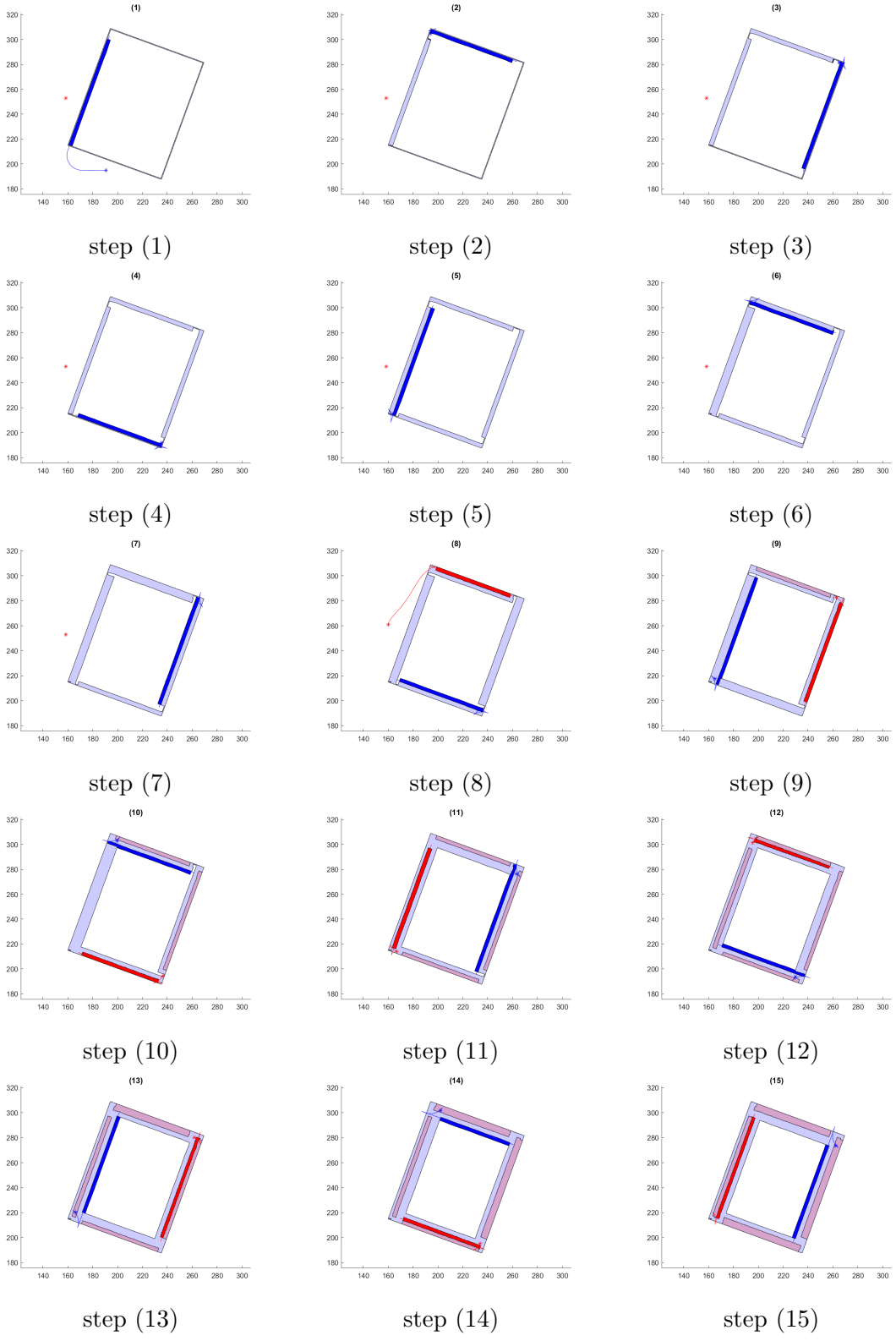
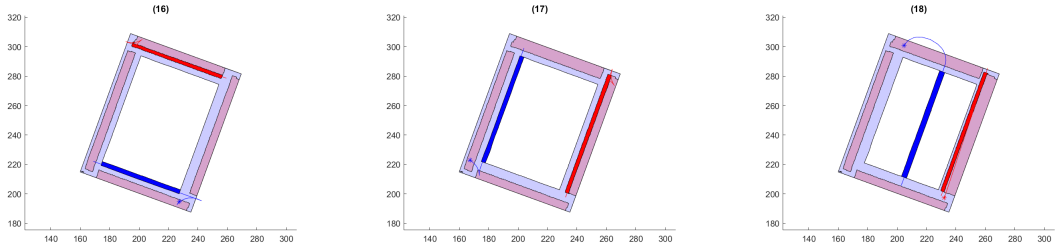
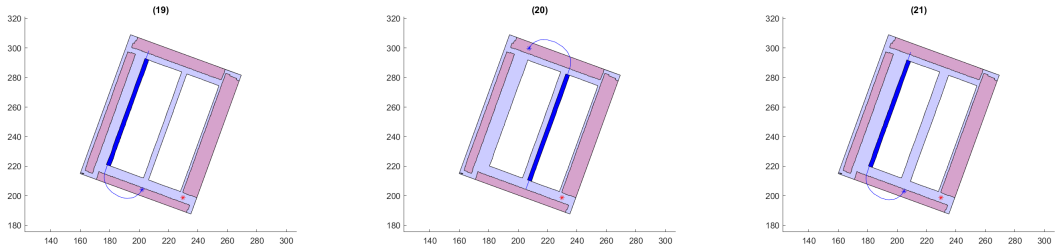


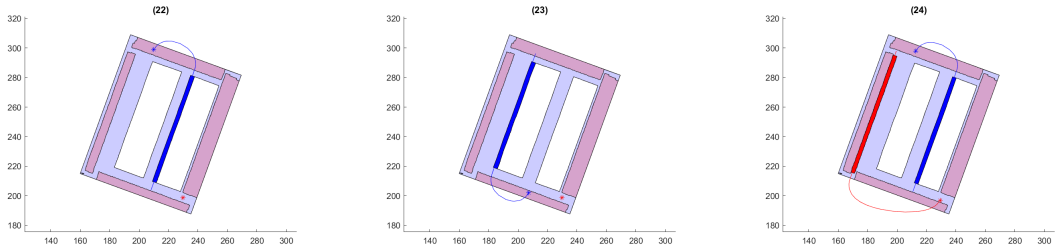
Figure A2: Progress of Algorithm A21 test drive steps 1–15.



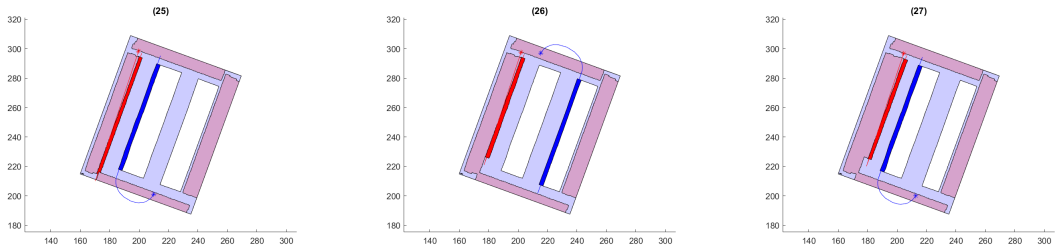
step (16) step (17) step (18)



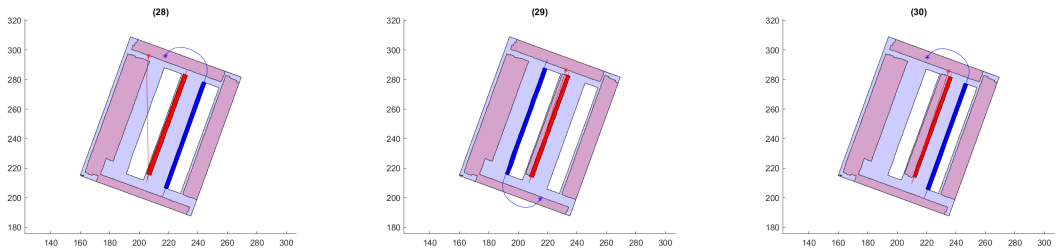
step (19) step (20) step (21)



step (22) step (23) step (24)



step (25) step (26) step (27)



step (28) step (29) step (30)

Figure A3: Progress of Algorithm A21 test drive steps 16–30.

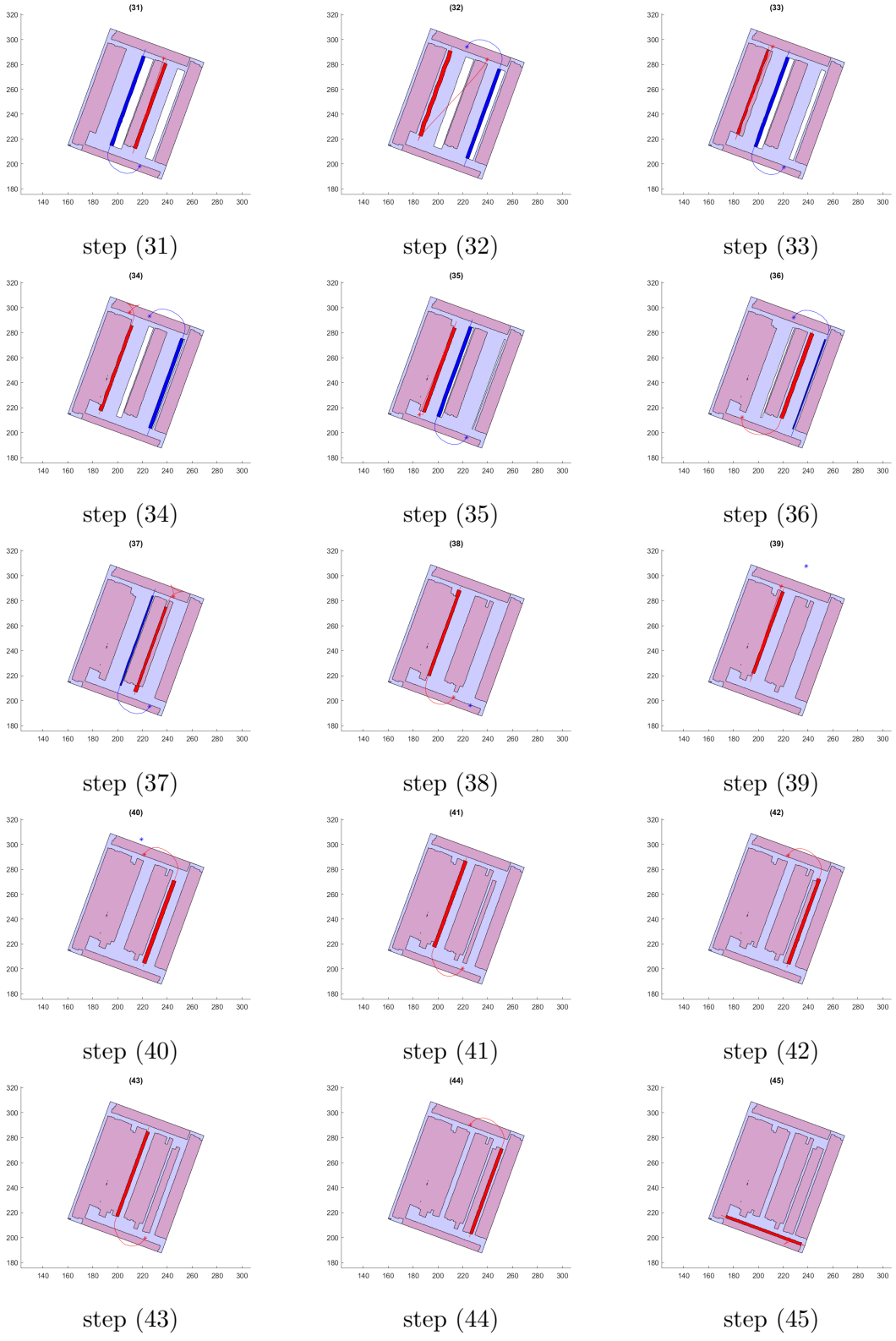


Figure A4: Progress of Algorithm A21 test drive steps 31–45.

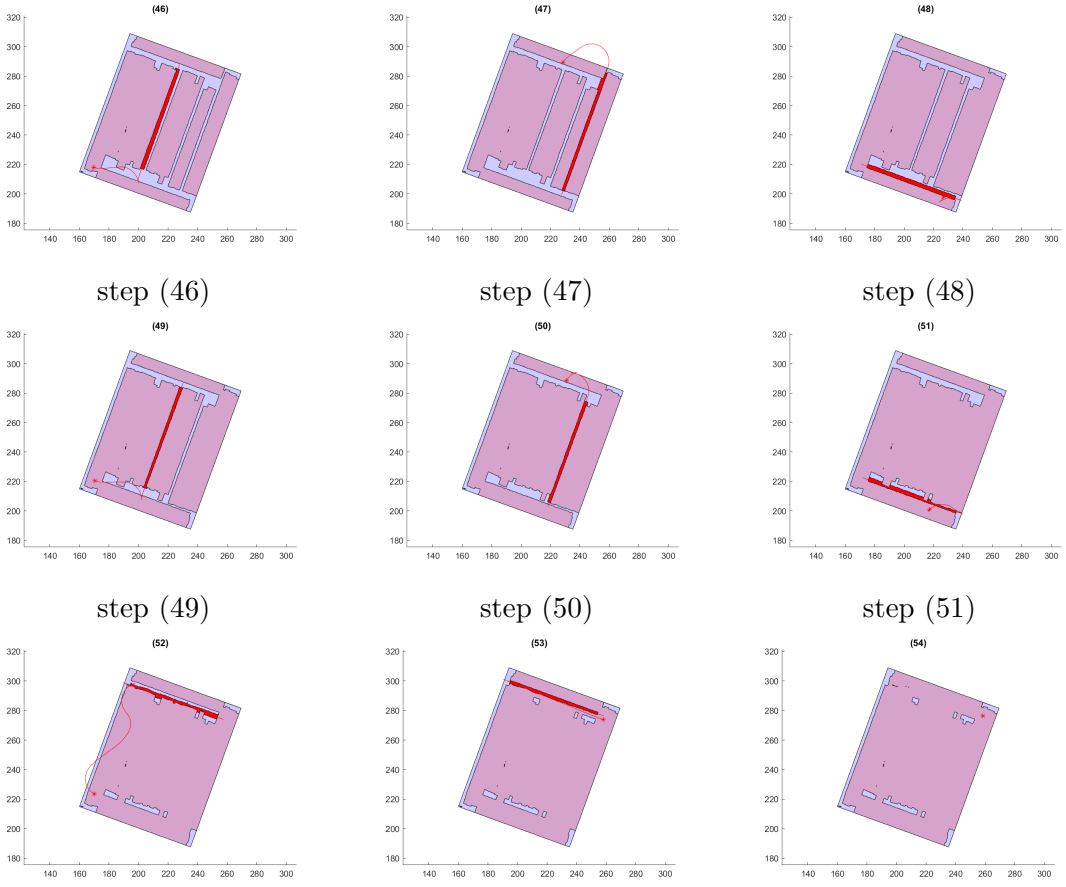


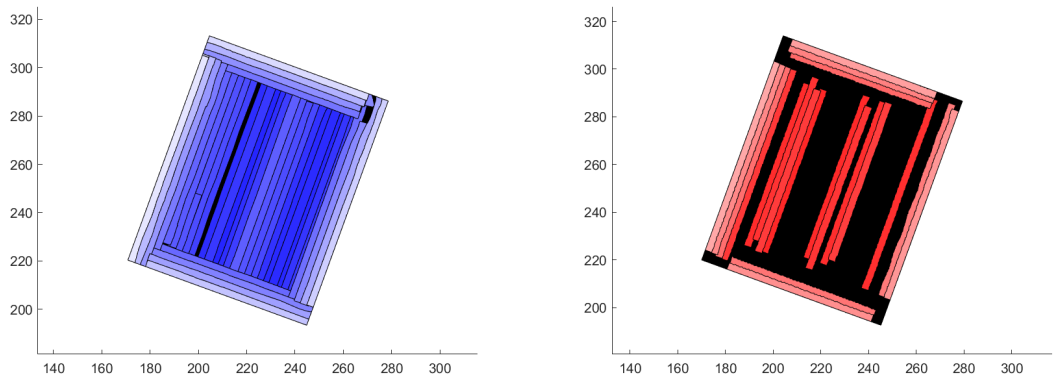
Figure A5: Progress of Algorithm A21 test drive steps 46–54.

Algorithm A31 Test Drive

In the figures the 1st vehicle data is plotted with blue color and the 2nd vehicle data is plotted with red color. The coordinates in all figures are in local coordinate frame where y-axis corresponds to YKJ northing and x-axis corresponds to YKJ easting.

The final coverage result is presented in figure A6. Color intensity indicates the temporal order of the swaths so that the most recent swaths are more intense in color. Black color indicates an area that was left uncovered. The coverage area is based on the vehicle GPS coordinates and the hitch level value which describes the vertical position of the tool.

The step by step progress of area coverage is presented in figures A7, A8 and A9. In chronological order, each step describes the result of one cycle of the algorithm and shows the previously covered area for both vehicles. A marker indicates the initial location of the vehicle during the step. A continuous line indicates the turn and a coloured polygonal region indicates the coverage area of the swath. If the vehicle does not move during the step only the marker is shown. The coverage area is based on vehicle GPS coordinates and the hitch level value which describes the vertical position of the tool.



(a) Area covered by 1st vehicle.

(b) Area covered by 2nd vehicle.

Figure A6: Covered area at the end of Algorithm A31 test drive.

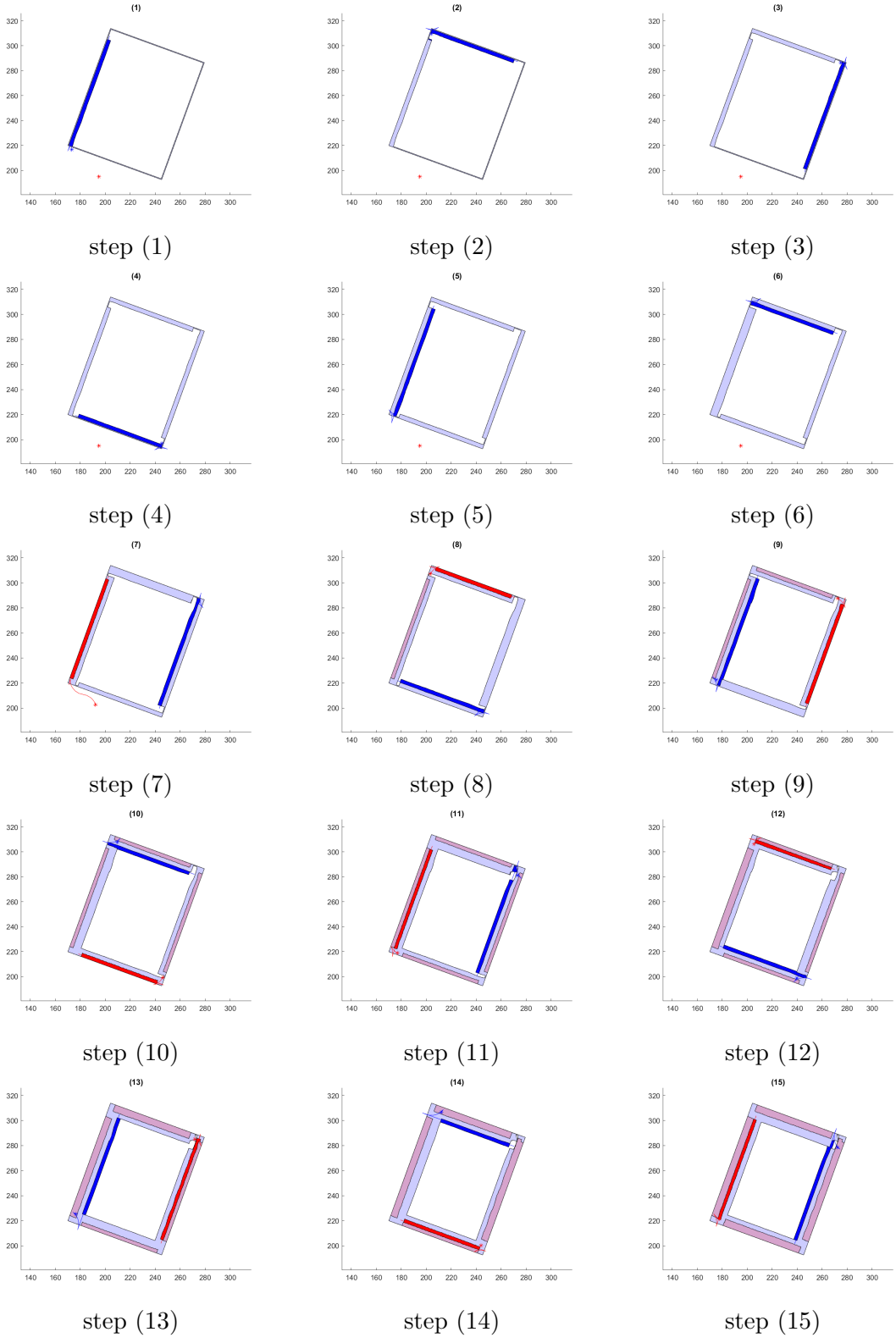
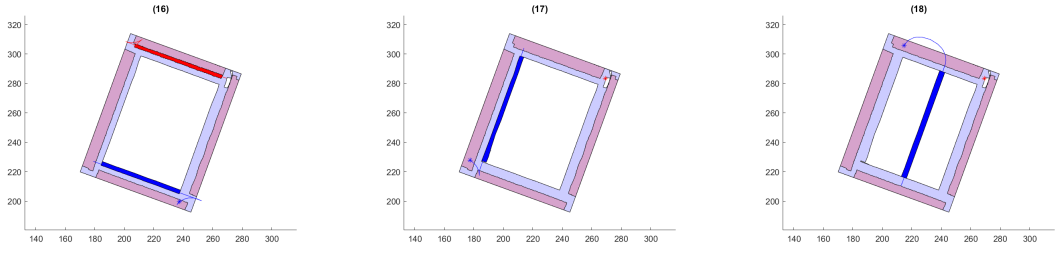
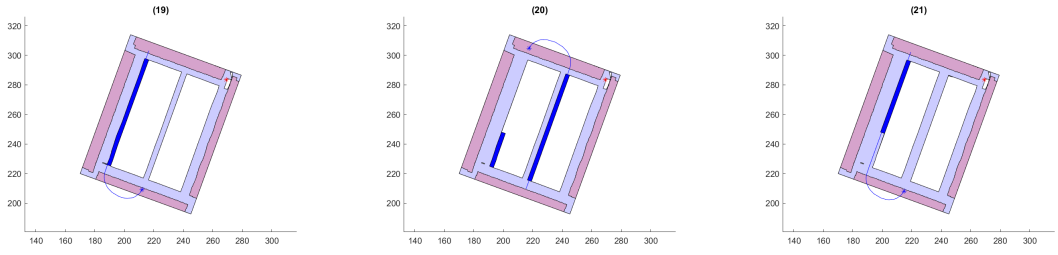


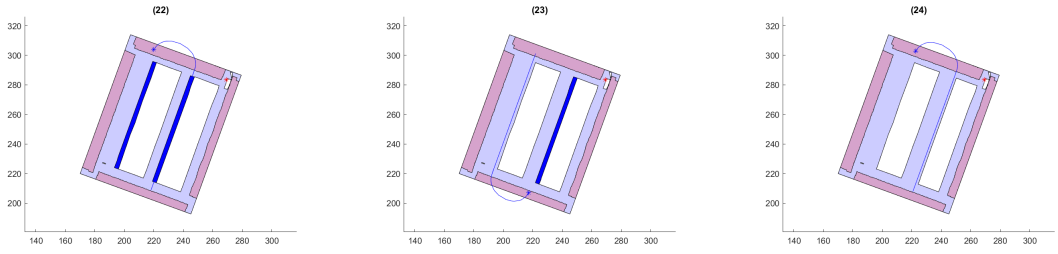
Figure A7: Progress of Algorithm A31 test drive steps 1–15.



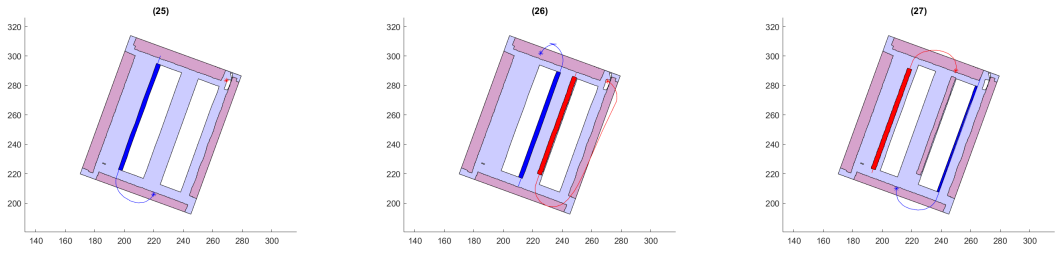
step (16) step (17) step (18)



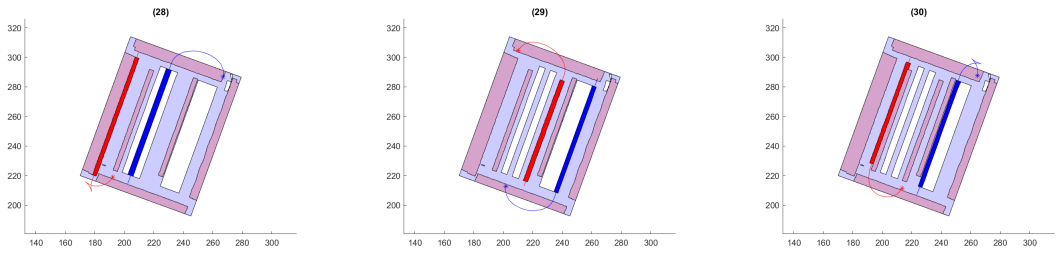
step (19) step (20) step (21)



step (22) step (23) step (24)

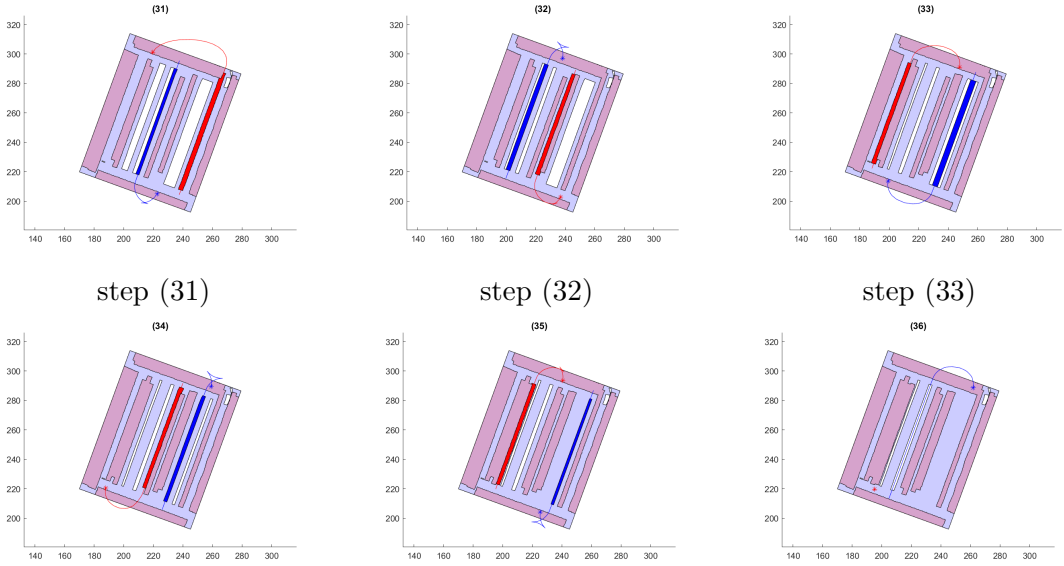


step (25) step (26) step (27)



step (28) step (29) step (30)

Figure A8: Progress of Algorithm A31 test drive steps 16–30.



step (31) step (32) step (33)

step (34) step (35) step (36)

Figure A9: Progress of Algorithm A31 test drive steps 31–36.

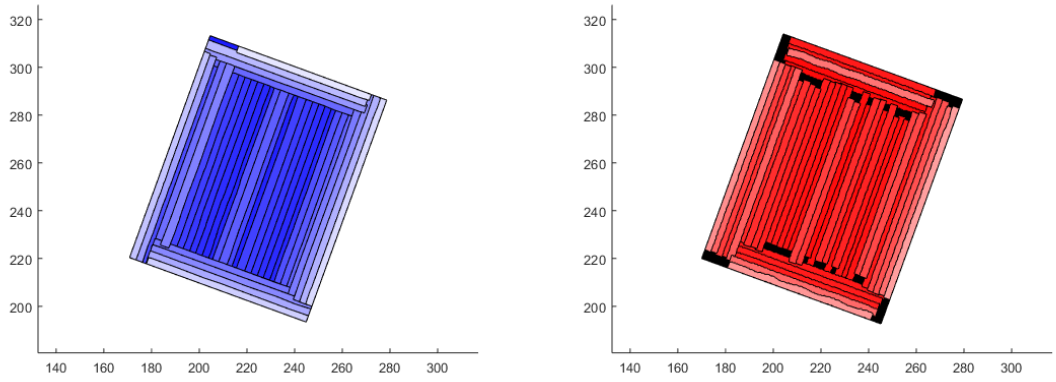
B Appendix

The details of Algorithm B test drive in Vihti, Finland on Fri the 28th of September 2018 are presented in the following figures. Algorithm B is described in detail in chapter 5.3. In the figures the 1st vehicle data is plotted with blue color and the 2nd vehicle data is plotted with red color. The coordinates in all figures are in local coordinate frame where y-axis corresponds to YKJ northing and x-axis corresponds to YKJ easting.

Algorithm B Test Drive

The final coverage result is presented in figure B1. Color intensity indicates the temporal order of the swaths so that the most recent swaths are more intense in color. Black color indicates an area that was left uncovered. The coverage area is based on the vehicle GPS coordinates and the hitch level value which describes the vertical position of the tool.

The step by step progress of area coverage is presented in figures B2, B3, B4 and B5. In chronological order, each step describes the result of one cycle of the algorithm and shows the previously covered area for both vehicles. A marker indicates the initial location of the vehicle during the step. A continuous line indicates the turn and a coloured polygonal region indicates the coverage area of the swath. If the vehicle does not move during the step only the marker is shown. The coverage area is based on vehicle GPS coordinates and the hitch level value which describes the vertical position of the tool.



(a) Area covered by 1st vehicle.

(b) Area covered by 2nd vehicle.

Figure B1: Covered area at the end of Algorithm B test drive.

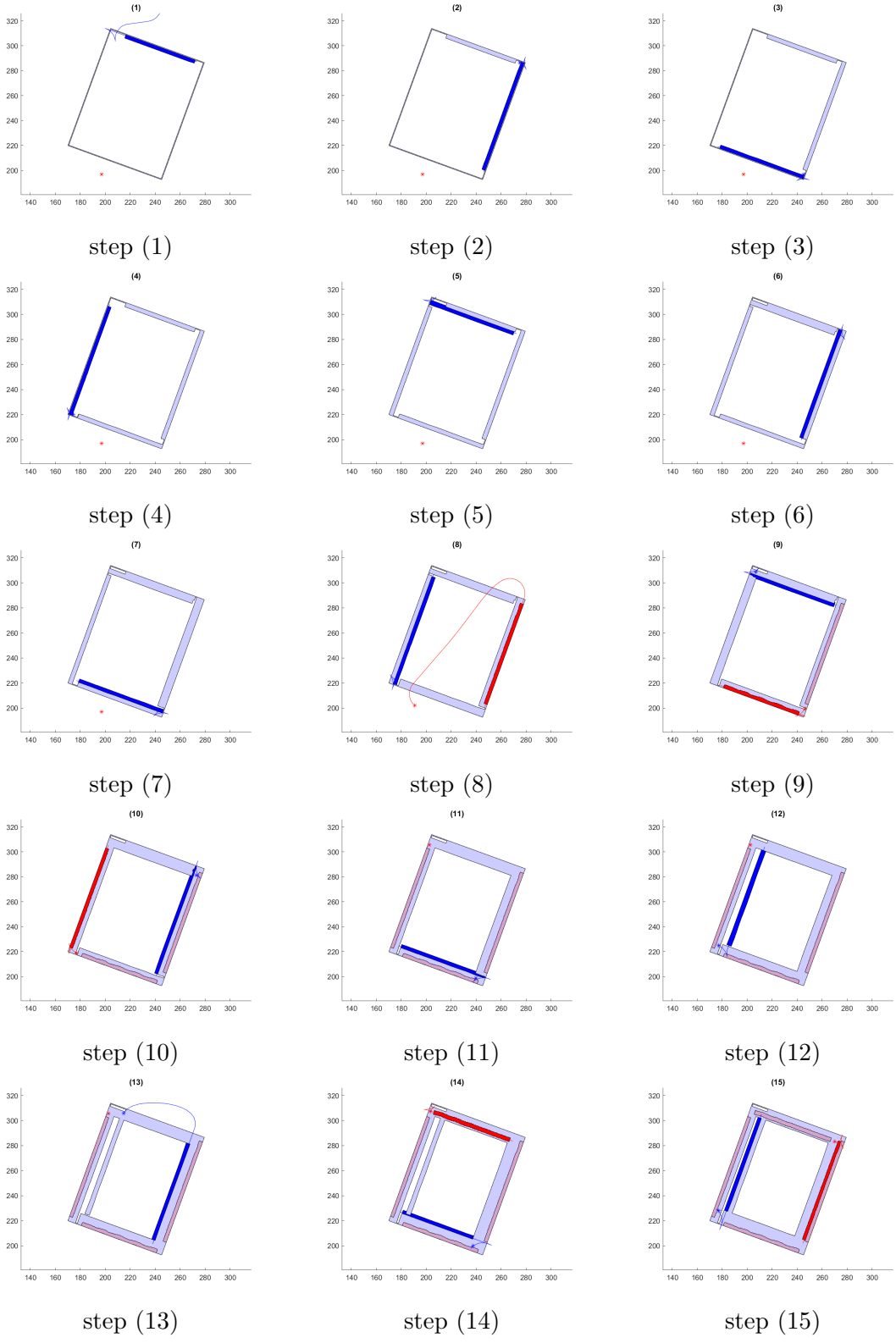


Figure B2: Progress of Algorithm B test drive steps 1–15.

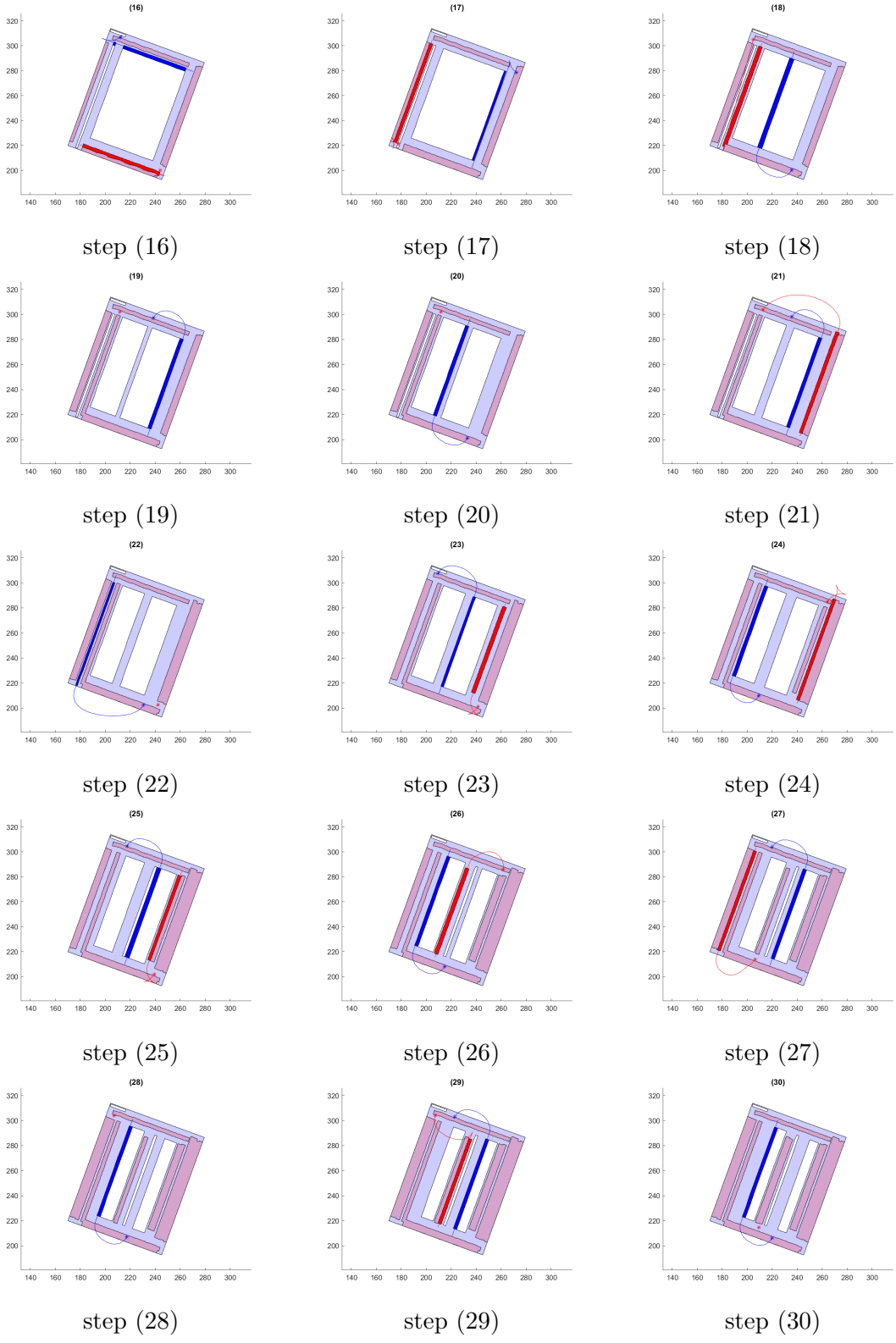


Figure B3: Progress of Algorithm B test drive steps 16–30.

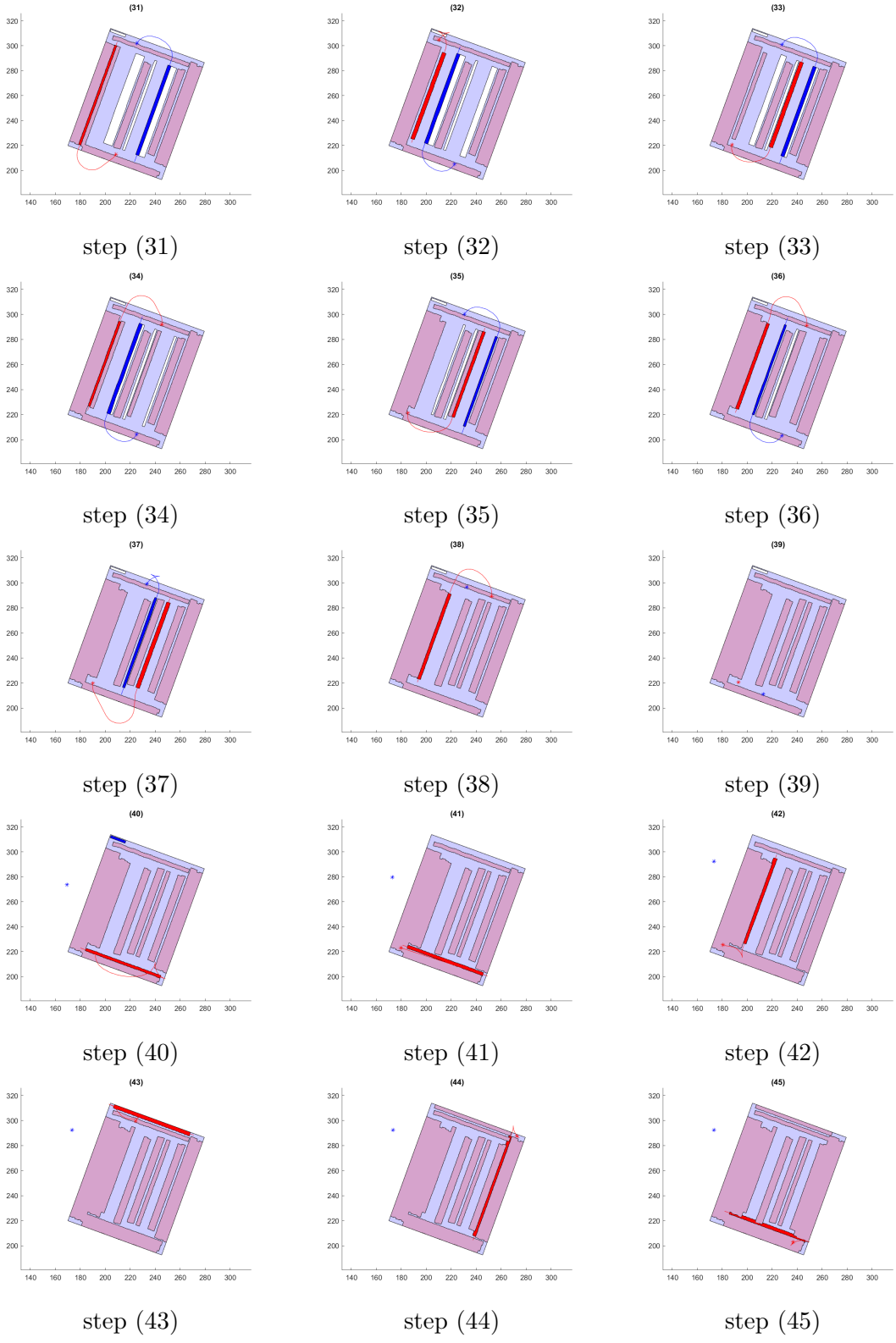


Figure B4: Progress of Algorithm B test drive steps 31–45.

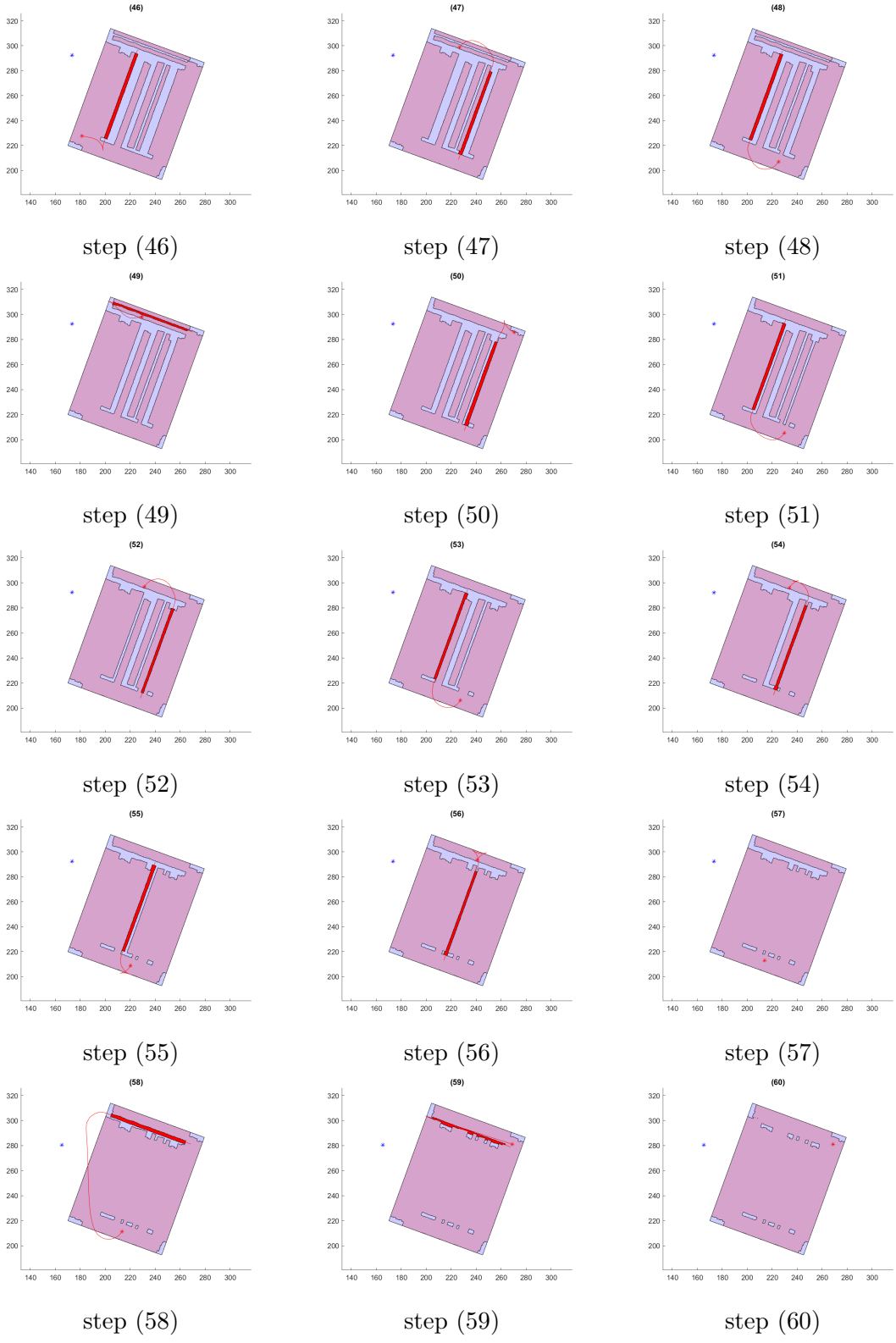


Figure B5: Progress of Algorithm B test drive steps 46–60.